NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CLOUD-BASED COMMUNICATIONS PLANNING COLLABORATION AND INTEROPERABILITY

by

Joni W. Pepin
Tarrell Giersch

June 2012

Thesis Co-Advisors: ManTak Shing
John Gibson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Cloud-based Communications Planning Collaboration and Interoperability		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Joni W. Pepin and Tarrell Giersch			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ___N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words) Communications planning is a key part of the Marine Corps operational planning process. The ability to design and analyze communication network plans efficiently and accurately has a direct impact on the ability of commanders to command and control actions on the battlefield. Portions of the current process of communications network planning for military exercises and operations in the Marine Corps are unnecessarily inefficient and susceptible to human error.

At the heart of network planning is the creation of accurate high-level diagrams that depict the details of the planned network topology for use in network installation, maintenance and operation. These diagrams are referenced at all levels in the planning, installation, operation and maintenance of the resultant communications architecture. Development and iterative refinement of these high-level network diagrams is a fragmented manual process. Despite the heavy reliance on network diagrams in the planning process, no software application currently exists that is designed specifically for their creation.

This thesis proposes a cloud-based application for communications planning. It describes the benefits achievable through automation, collaboration and application interoperability, and provides recommendations for development of such a system. It concludes by presenting an implementation of these recommendations via a proof of concept application.

14. SUBJECT TERMS Communications Planning, Cloud Computing, Real Time Collaboration, Interoperability, Automation			**15. NUMBER OF PAGES** 151
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CLOUD-BASED COMMUNICATIONS PLANNING COLLABORATION AND INTEROPERABILITY

Joni W. Pepin
Captain, United States Marine Corps
B.S., University of South Carolina, 2002

Tarrell Giersch
Major, United States Marine Corps
B.S., Hawaii Pacific University, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2012

Authors: Joni W. Pepin
Tarrell Giersch

Approved by: Man-Tak Shing
Thesis Co-Advisor

John Gibson
Thesis Co-Advisor

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Communications planning is a key part of the Marine Corps operational planning process. The ability to design and analyze communication network plans efficiently and accurately has a direct impact on the ability of commanders to command and control actions on the battlefield. Portions of the current process of communications network planning for military exercises and operations in the Marine Corps are unnecessarily inefficient and susceptible to human error.

At the heart of network planning is the creation of accurate high-level diagrams that depict the details of the planned network topology for use in network installation, maintenance and operation. These diagrams are referenced at all levels in the planning, installation, operation and maintenance of the resultant communications architecture. Development and iterative refinement of these high-level network diagrams is a fragmented manual process. Despite the heavy reliance on network diagrams in the planning process, no software application currently exists that is designed specifically for their creation.

This thesis proposes a cloud-based application for communications planning. It describes the benefits achievable through automation, collaboration and application interoperability, and provides recommendations for development of such a system. It concludes by presenting an implementation of these recommendations via a proof of concept application.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AJAX	Asynchronous Javascipt and XML
CCP	Common Communication Picture
COMMCON	Communications Control
COTS	Commercial Off-the-Shelf
DISA	Defense Information Systems Agency
GOTS	Government Off-the-Shelf
GSORTS	Global Status of Resources and Training System
GUID	Global Unique Identifier
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JCSS	Joint Communication Simulation System
MAGTF	Marine Air Ground Task Force
MARCORSYSCOM	Marine Corps Systems Command
MCPP	Marine Corps Planning Process
MCWP	Marine Corps Warfighting Publication
MEF	Marine Expeditionary Force
SaaS	Software as a Service
SOA	Service Oriented Architecture
SPE	Systems Planning and Engineering
SPEED	Systems Planning Engineering and Evaluation Device
SYSCON	Systems Control
TCP/IP	Transmission Control Protocol/Internet Protocol
TECHCON	Technical Control

W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. We would like to express our sincere appreciation to our advisors, Dr. Man-Tak Shing and Professor John Gibson, for their guidance, mentorship and patience throughout the time spent on this thesis. Your input and “rudder steer” have been an invaluable part of the entire experience.

Joni Pepin

I hope to one day be able to somehow reciprocate the level of support and understanding that I have received from my wife and kids over the last year. This has truly been a team effort, and for that, I am eternally grateful.

I would also like to thank Captain Le Nolan, USMC, and Captain Drew Abell, USA, for their input in the software engineering process. Your contributions are greatly appreciated.

Tarrell Giersch

Special thanks to my friends and colleagues, especially group members of: Team Searching, Team Wolfpack, and Team Avengers for sharing their experience and invaluable assistance.

Finally, I would also like to thank my family for the support they provided me throughout my life and in particular, I would like to acknowledge my mother, Carolyn Giersch, whose love and encouragement continues to guide and motivate me each and every day.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Communication enables and supports command and control therefore communications planning must be detailed, accurate, and flexible enough to respond to a rapidly changing battlespace. The relationship between command and control and communication is complex. For a Commander, command and control is the critical process by which mission objectives are delineated, allocated to subordinates, and monitored for compliance. Information is critical to this process because it informs everyone throughout the chain of command of their role and responsibilities and develops their situational awareness, allowing better decisions to be made, and better execution of tasks. Information is essential to a more complete cycle of assessment and adjustment. This complete cycle, to include information flow, is the basis of command and control.

The main element that adversely affects command and control is uncertainty; enabling commanders to better deal with this uncertainty is the goal of planning. The flexibility and agility of the planning process enables planners to adapt to changes in either the mission objectives or the operational environment. Efficient information exchange contributes greatly to achieving this flexibility and agility in the planning process, allowing the commander to focus on more important operational opportunities or constraints. Communications planning, like other forms of planning, is sequential, concurrent, repetitive, distributed, and continuous. It runs concurrent to, and in conjunction with, the overall planning effort.

Development of the communications plan is based on the overall concept of operations, and directly supports mission accomplishment. Like all planning, communications planning should not be conducted without consideration for, and interaction with, other planning domains (such as supply, logistics, and

maintenance). The inputs and outputs of communication planning derive from, and are fed into, the overall plan, respectively.

The current process of communications network planning for military exercises and operations in the Marine Corps are unnecessarily inefficient and susceptible to human error. Software applications currently used by Marine Corps communications planners and operators can be divided into two distinct categories: network modeling and simulation, and network monitoring. Network modeling and simulation software takes input from the user, such as geographical terrain data and transmission system specifications, and predicts the ability of equipment to transmit effectively between operation locations. Network monitoring software assists in the optimization of deployed networks, maintains real-time network status, and assists in troubleshooting network problems. The applications currently used by the Marine Corps communications network planners and operators to perform these functions represent multiple software tools throughout the development, analysis, implementation, and monitoring of communications networks. Unfortunately, many of these tools are not interoperable, thereby compounding or confounding the planning and monitoring process.

B. PROBLEM STATEMENT

There is currently no work directly related to the development of a cloud-based application as a means to improve collaboration between Marine Corps communications planners and operators, decrease communications plan errors, and reduce generation time in the planning process. While there exists powerful modeling and simulation applications designed to determine if a network link can feasibly work and network monitoring tools that tell the user if a link will actually work as expected in real world operation, there are no applications that assist in the creation of the plan on which these two rely. Additionally, no application exists that provides a conduit through which the existing planning and monitoring tools can exchange information, the absence of which results in duplication of

effort, increased vulnerability to human error, and a slower planning cycle. The goal of the proposed research is to demonstrate the potential utility of a cloud-computing Software as a Service (SaaS) application to improve processes and products in the field of Marine Corps communications planning through automation, collaboration, and interoperability of communications planning and monitoring tools.

C. THESIS ORGANIZATION

Chapter II provides some background on the Marine Corps Planning Process (MCPP) and how it relates to current network planning processes, specifically in the areas of automation, collaboration, and interoperability. It introduces the idea of using the Software as a Service (SaaS) model to develop a cloud-based communications planning application, and it discusses the benefits of the proposed application. Additionally, it identifies the stakeholders and their goals, clearly outlines the objectives and success criteria, and introduces the key use-cases required to meet the objectives. Chapter III identifies the requirements necessary for collaboration and coordination in communications planning, the tools currently used to achieve them, and the shortfalls inherent to the planning process. It specifically addresses designing software for collaboration between communications planners in a cloud environment and proposes the software design patterns best suited to accomplish this task. Chapter IV covers how a more efficient planning process can be achieved by removing some of the redundant manual work that results from lack of interoperability between communications planning and monitoring software systems. It identifies some industry accepted, enabling technologies that can be leveraged to provide a means to achieving interoperability. It presents a method of implementation that enables interoperability between existing and future systems through identification and sharing of redundant communications planning and monitoring system entities and information sets. Chapter V provides an overview of the resultant proof-of-concept application, the Common Communication Picture (CCP). It demonstrates the ability of

communications planners to collaborate in near-real time in the communications planning process and enables interoperability between the CCP application and other existing network planning and operation tools. Chapter VI presents a conclusion, identifies contributions, and recommends areas for future work.

II. BACKGROUND

A. THE MARINE CORPS COMMUNICATION PLANNING PROCESS

Marine Corps Warfighting Publication (MCWP) 5–1 defines the three tenets of the Marine Corps Planning Process (MCPP) (Figure 1) as: "top-down planning, single-battle concept, and integrated planning" [1]. Top-down planning places the responsibility of planning on the commander, who provides guidance to planners through his concept of operations. As part of the Problem Framing phase of operational planning, communications planners analyze the commander's intent, as well as all specified, implied and essential tasks, and develop a mission statement based on information such as task organization and resource availability. Led by their respective G-6/S-6, communications planners use this initial analysis in the engineering of communications networks that support the possible courses of action derived in the Course of Action Development phase of the MCPP.

The single-battle concept is derived from the observation that all actions in the battle space have the ability to affect other areas or functions in the battle space. This is equally true for tactical networks, which grow and transform dynamically as an operation evolves. Changes in one aspect of the network can have far-reaching effects on other aspects of the network. For example, movement of a single satellite link from one site to another can have a direct impact on the provision of communications services at each site, and thus affects a commander's ability to command and control.

Communications planning runs concurrent to, and in conjunction with, the overall planning effort. Development of the communications plan is based on the overall concept of operations, and in support of mission accomplishment. Like all planning, communications planning should not be conducted without consideration for, and interaction with, other planning considerations (such as supply, logistics, and maintenance). Typically led by the G-6, communications

planning is accomplished through application of the MCPP by communication units. The inputs and outputs of communication planning are derived from, and fed into, the overall plan, respectively.

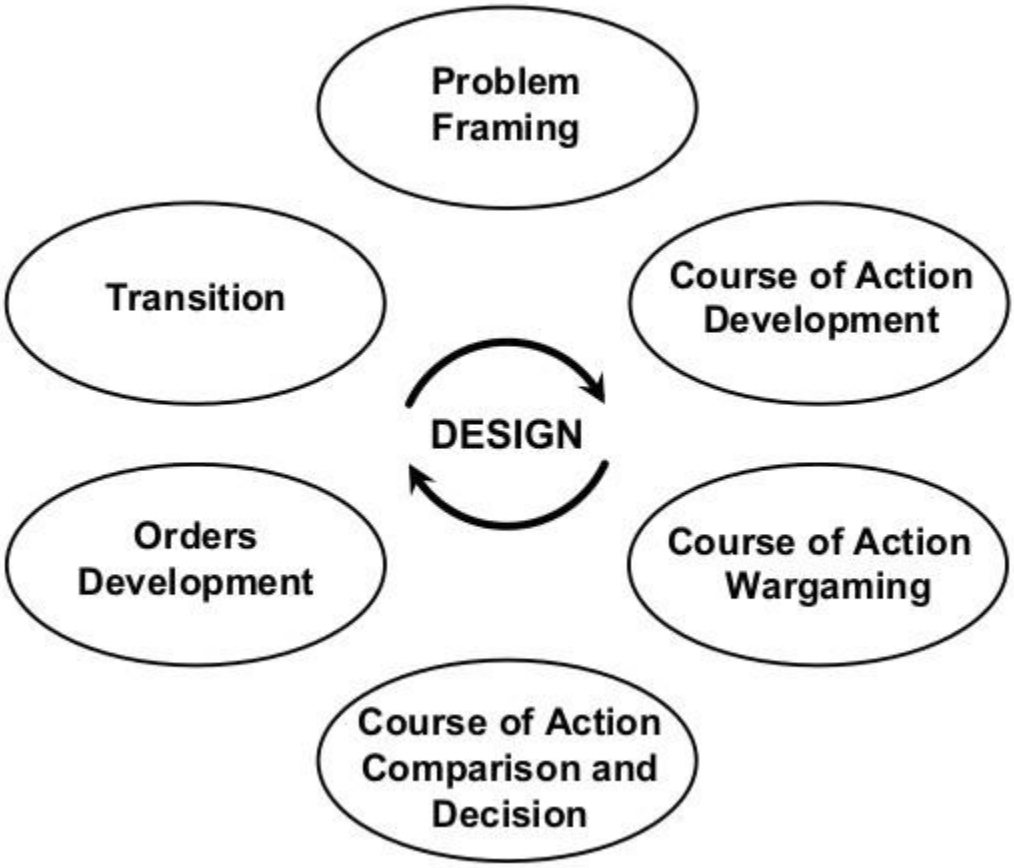


Figure 1. The Marine Corps Planning Process (MCPP). From [1]

Portions of the current process of communications network planning for military exercises and operations in the Marine Corps are unnecessarily inefficient and susceptible to human error. At the heart of network planning is the creation of accurate high-level diagrams that depict the details of the planned network topology for use in network installation, maintenance and operation. Development and iterative refinement of these high-level network diagrams is a fragmented manual process, wherein multiple planners work individually on their respective subsections, which include both subordinate and adjacent commands and, within each command, all required sub-network diagrams. Planners typically

use Microsoft Visio or PowerPoint to create and update network diagrams. These diagrams are aggregated by a central figure (usually the lead planner at higher headquarters, such as the Marine Expeditionary Force (MEF) G-6) and distributed accordingly. In the event of an error (such as an erroneous bandwidth allocation) or a change in network infrastructure, the diagrams are again edited, aggregated, and redistributed. Every piece of the network diagram, from the pictures representing hardware, to the text annotating available bandwidth, is entered manually. This increases the likelihood of human error. Through multiple planning meetings, the errors are flushed out, and the diagrams are cycled as before, until they are eventually finalized. Additionally, because there are multiple versions of the plan stored and distributed throughout the planning and implementation process, it is commonplace to experience network problems due to installers and maintainers using outdated versions of network diagrams when installing and troubleshooting the network.

In addition to following the tenets of the Marine Corps Planning Process, communications planning, like other forms of planning, is “sequential, concurrent, repetitive, scalable, and continuous” [2]. It stands to reason, then, that the software tools used in the planning process should be designed in a manner that, understanding the dynamic nature of communications planning, enhances a planner’s efficiency and accuracy, while minimizing the repetitive nature of the process through the use of automation whenever feasible.

B. THE CURRENT STATE OF NETWORK PLANNING SOFTWARE

Software applications currently used by Marine Corps communications planners can be divided into two distinct categories: network modeling and simulation, and network monitoring. Given information such as geographical terrain data and transmission system specifications, modeling and simulation applications can predict the ability of equipment to transmit effectively between two locations. Once a network has been installed, network monitoring tools are used by planners for not only maintaining real-time network status, but to also

assist in troubleshooting efforts, and in the identification of possible network optimizations (for instance, identifying under and/or over-utilized network links, and possible alternate routes).

In the Marine Corps, the modeling and simulation tool used by System Planning Engineer Officers (SPEOs) is known as the Systems Planning Engineering and Evaluation Device (SPEED) [3]. SPEED is a Government Off-the-Shelf (GOTS) program maintained by Marine Corps Systems Command (MARCORSYSCOM) that is used by Marine Corps communications planners for modeling and simulation of transmission systems and frequency spectrum analysis. According to MARCORSYSCOM, SPEED has been “selected by other services/agencies as the best (Radio Frequency) RF link engineering tool,” and is used by the Army, Air Force, and other Federal Agencies. SPEED was initially fielded in 1990 by the Marine Corps as a communications planning tool for use in garrison and tactical environments. It provides network planners with the ability to perform transmission link analysis and engineering. In addition to link analysis, SPEED is capable of producing radio guard charts, satellite access request forms, and communications equipment cut-sheets. SPEED is a modular system, maintained as a program of record by MARCORSYSCOM, and is updated regularly to include all current communications transmission systems in use in the Marine Corps. It is intended for use in all elements of the MAGTF by the G-6 and respective communications unit (communications battalion, squadron and company) systems planning engineers and other communications planners. Collaboration support is built-in via file import and export, providing higher echelons of command the ability to combine plans from lower echelons. SPEED requires the Microsoft Windows delete all trademark symbols operating system, and can be used as either a standalone application or as a C2PC Injector.

Within the Department of Defense, the Joint Communication Simulation System (JCSS) [4] is a desktop application similar to SPEED that “provides modeling and simulation capabilities for measuring and assessing the information flow through the strategic, operational, and tactical military communications

networks” [4]. Formerly called NETWARS, JCSS was initiated in 1996 by the director of C4, J-6, based on his concern that the contingency systems developed by the J-6 planners could “collapse” in a fully operational environment. According to the Defense Information Systems Agency (DISA), who is responsible for maintaining the JCSS program, JCSS is the current standard for modeling military communications systems within the Department of Defense. While SPEED primarily focuses on the modeling and simulation of radio-frequency-based transmission equipment, JCSS takes the next step by also modeling back-end network equipment, such as commercial and tactical data and voice/telephony. It also provides the ability to run simulations and conduct load testing of simulated networks.

There are several commercial network monitoring tools available for use on military networks. Of these, Solarwinds’ Orion [5] and Ipswitch’s WhatsUp Gold [6] are two examples of applications that are currently in use on both training and operational Marine Corps tactical networks. They provide the real-time connectivity status of the network through the use of SNMP and IP monitoring. Unlike modeling and simulation tools which are used early in the planning process to determine if communication is feasible between two sites (i.e., what *can be* done), network monitoring tools are utilized after the network has been installed. In contrast to SPEED and JCSS, which are used primarily by the communication unit’s systems planning and engineering (SPE) section, tools such as Orion and WhatsUp Gold are used by systems control (SYSCON) and technical control (TECHCON) to provide the ability for the real-time monitoring of live networks (i.e., what *is being* done).

Working as mutually symbiotic functional areas of communications control (COMMCON), the operational SYSCON uses the network diagrams developed by the SPE section, in concert with the real-time information provided by tools such as Orion and WhatsUp Gold, in the real-time management of the operational network. For example, if while monitoring the network the SYSCON watch officer is alerted by Orion that a link between two sites has failed, he can

quickly reference the network diagrams produced during network planning to determine what effect the outage has on the network as a whole by validating redundant communications links, calculating available bandwidth, and possibly determining the source of the problem. If required in response to the outage, the SPE section will modify the network topology (and as a result, the network diagrams), provide all functional areas the updated plan, and oversee the implementation of any changes.

C. ENABLING A MORE EFFICIENT PLANNING PROCESS

A lot of time has been put into powerful modeling and simulation applications designed to determine if a network link could feasibly work, and network monitoring tools that tell the user if a link actually works as expected in real world operation. However, no applications are available to assist in the creation of the plan on which these two rely. Additionally, no application provides a conduit through which the existing planning and monitoring tools can exchange information, resulting in duplication of effort and a slower planning cycle. The need for such an application, coupled with the benefits provided by a cloud-based system and the mandate to centralize DoD systems, leads to the logical conclusion that research should be conducted to determine the requirements of a cloud-based software solution, and to outline how such a system would be engineered in light of these requirements.

In order to create a software solution that enables a more efficient communications planning process, the proposed software must address three key areas: automation, collaboration between communications planners, and interoperability of communications planning and monitoring tools.

1. Automation

It is understood that the responsibilities of the communications planners do not end at the completion of the pre-operation planning phase. As a function of the command and control of the network, in addition to network planning, the controlling communications agency is also responsible for the installation,

operation, and maintenance of the network, and all that is entailed in each of these areas. In the Marine Corps, these COMMCON responsibilities are typically divided into the three functional areas already mentioned: systems planning and engineering, operational systems control, and technical control [7]. In addition to the network modeling and analysis completed by the network planners using tools like SPEED, some of the most important products to come out of the planning process are the diagrams depicting the network topology (see Appendix A). Developed based on equipment availability and capabilities, and in support of the chosen course of action, these high-level diagrams are keystone documents used in all successive phases of installation, operation and maintenance. Similar to the manner in which maps and tactical overlays can be used to provide a common operational picture, network diagrams provide a common communications picture by ensuring a common understanding of the "big picture" by all communications elements.

The task of automating the Marine Corps communications planning process is still in its infancy. As part of the overall military planning process at all levels of warfare, from tactical to strategic, the ability to design, analyze and monitor communications networks in the most efficient way has a direct impact on the military commander's ability to exercise command and control on the battlefield.

Despite the importance of these network diagrams in the planning, installation, operation and maintenance of communications networks, there is no available software *specifically* designed to assist communications planners in their creation. This thesis defines a software system that is designed to enable collaboration between planners, and to be used in concert with, and as a bridge between, the two previously identified categories of network software. One of the goals of the proposed system is to enable a more efficient communications planning work-flow through the automation of repetitive tasks in network design (such as bandwidth allocation and verification, equipment compatibility, etc.), and by eliminating redundant tasks involved in the use of multiple disparate planning

tools. Automation will minimize the possibility of human error in creation of the high-level network diagrams, and the transfer of information between systems.

2. Collaboration

The second goal of the proposed system is to enable a more efficient communications planning work flow through provision of the near real-time collaboration between planners that a cloud-based solution could provide. As a simple illustration of one of the possible benefits of a cloud-based communications planning application, consider the previous example, wherein the SPE section makes changes to the active network topology in response to some type of network outage. If, in making the required changes to the network topology, the network planners were able to use a cloud-based application for the production of network diagrams, it would enable the near real-time availability of any updated network plans not only to the local SYSCON and TECHCON, but to all other units operating in the battle space (keeping in mind the single-battle concept). Due to the fluid and dynamic nature of war, real-time updates of the network plan to all concerned parties could lead to minimized down-time, and thus, minimize the effect that the network outage has on the commander's ability to command and control his forces throughout the battle space. Enabling a common communications picture would benefit not only those units in the area of operations, but also any unit that has a need to know the network topology. An apt example of this is a unit training for eventual assumption of network control. Having access to the most up-to-date network diagrams provides the incoming unit the ability to conduct scenario-based training on similar network topologies prior to deployment, and could result in a more seamless transfer of control.

3. Interoperability

The applications described above are representative of the multiple software tools used by the Marine Corps throughout the planning process in the development, analysis, implementation and monitoring of networks. Like the planning process itself, these software tools are fragmented, with limited ability to

exchange data between them. While information exchange is technically possible due to the ability for most available tools to import and export information using the Extensible Markup Language (XML) standard, the lack of standardization across applications results in incompatibilities when attempting to seamlessly move and utilize the information from each other. The result is the duplication of effort and enhanced vulnerability to human error as network information is migrated manually between the different software systems. For example, a network topology modeled by system planners using SPEED must be manually entered into Orion, effectively doubling the amount of time and effort required. In order to eliminate this redundancy, the proposed system would serve as a bridge between the tools used in design, modeling, and monitoring phases of communications planning (Figure 2).

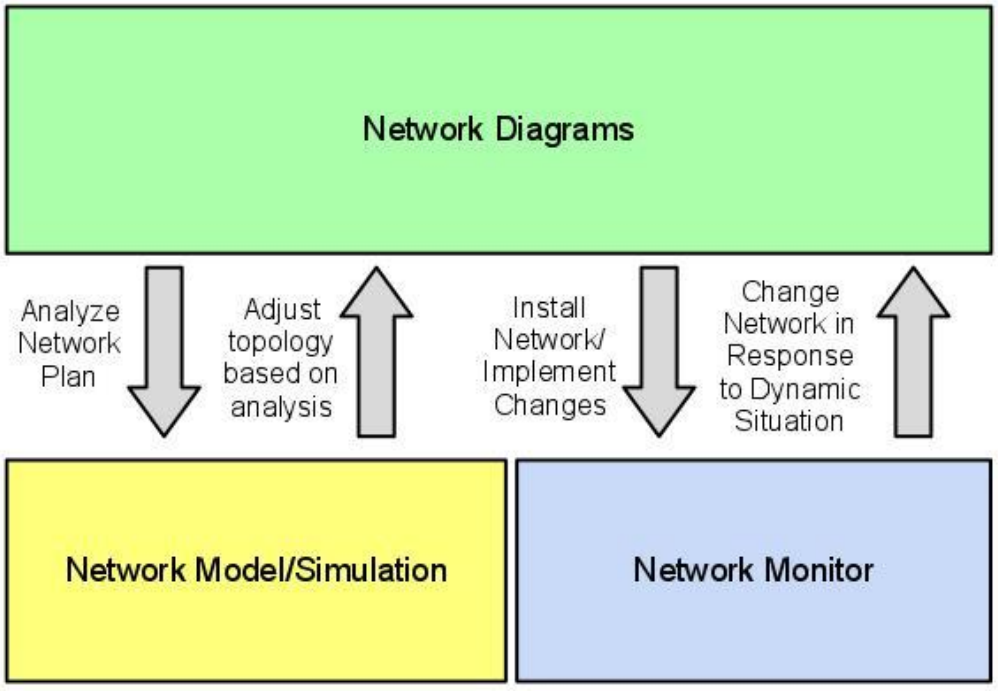


Figure 2. Relationship between network diagrams and network applications

4. Access To Centralized Equipment Specification and Availability Information

A common requirement among all previously mentioned planning tools, including the one proposed in this thesis, is information regarding the technical specifications and availability of communications equipment. Currently, each existing application maintains equipment capability information in flat files on the user's system, which are updated as capabilities change or new equipment is acquired. It is incumbent on the user to ensure that the information being used is the most current. As opposed to the flat file systems currently in use, the proposed system model would access a centralized equipment database, designed to be universally accessible by all communications planning systems. Storing equipment information in a centralized, universally accessible manner would enable the instant update of information for all users, and delegates the responsibility of maintaining current technical capability data to the database maintainer. The centralization of this information would further assist in the minimization of human error in the planning process.

In addition to equipment capabilities and limitations, information regarding the availability of equipment could also be incorporated to provide planners instant knowledge of available resources, further speeding up the planning process. The reporting of equipment status and availability is already accomplished through the Joint Staff directed requirement for all units to use the Global Status of Resources and Training System (GSORTS) [8]. Units are required to update this status every 30 days (or more frequently as required). This information is used for crisis response planning and deliberate or peacetime planning [9]. Integration of these systems with the equipment specification database would provide a robust capability for future communications planning.

D. CLOUD COMPUTING

On February 8, 2011, U.S. Chief Information Officer, Vivek Kundra, published the "Federal Cloud Computing Strategy" [10], in which he stated that

the “Federal Government’s current Information Technology (IT) is characterized by low asset utilization, a fragmented demand for resources, duplicative systems, environments which are difficult to manage, and long procurement lead times.” The document outlines the role of cloud computing in addressing these deficiencies. As one of the largest sectors, and expenses, of the Federal Government, the Department of Defense has a vested interest in capitalizing on the efficiencies afforded by migrating processes to a cloud environment. By utilizing the Software as a Service (SaaS) model, a cloud-based communications planning application may provide an efficient, centralized alternative to the current communications network planning tools and processes used in the Marine Corps and throughout the Department of Defense.

E. BENEFITS OF PROPOSED SYSTEM

Due to the dynamic nature of communications planning, and the responsibility of maintaining the expeditionary nature of Marine Corps operations, it is important that the tools used in the communications planning process are easily accessible, globally available, and platform independent. Implementation of such an application would allow communications planners to focus more on the product—a communications plan that best enables command and control - and less on the tool.

The proposed application, known as the Common Communication Picture (CCP) network planning tool, would provide the features listed below. While each of the individual features listed may not be mutually exclusive to this application, their aggregation in a single communications planning system is:

- Universally accessible, with no client install required
    - Interaction with server through web browser
    - Operating system and web browser independent
- Centralized planning environment
    - Multiple users able to edit existing plan concurrently

    - Updates viewable immediately by others
- Automated error checking
    - Hardware and configuration compatibility
    - Proper resource and bandwidth allocation
- Version control
    - Save and view multiple versions of plan during planning process
    - Serve as a planning references for future networks
    - Access to previous versions of a plan could be extended to serve as a rolling network change log for live networks
- Templating
    - Plan can be saved for use in future exercises or operations
- Role-based permission hierarchy
- Portability
    - The ability to install separate instances on multiple networks (e.g. NIPR and/or SIPR)
    - Use of standard software means that existing server architectures can be used
- System administration of plans and user accounts

F. SYSTEM STAKEHOLDER DESCRIPTION

1. Demographics

Communications network Planning, Installation, Operation and Maintenance (PIOM) is a highly technical field. Typical system users will be trained in multiple communications specialties. PIOM of a complete network that encompasses all critical capabilities (including data, telephone, satellite, single-

channel radio, multi-channel radio, multiplexing and encryption assets) involves the collaboration of each of these specialty areas in order to produce a robust and reliable communications infrastructure. The beneficiaries of this process are the leaders and personnel who rely on these networks in the execution of their assigned tasks.

2. Stakeholder (Non-users) Summary

The primary non-user stakeholders are unit commanders and personnel at all levels. Unit commanders require reliable and on-demand communications in order to command and control actions in the battle space. Likewise, all unit personnel rely on communications for internal and external coordination, and as a backbone over which a common operational picture is established and maintained.

Non-user stakeholders also include:

- External organizations and agencies that rely on the communications infrastructure in the execution of their respective tasks.
- The Defense Information System Agency (DISA), as the approving authority for a large portion of the communications architecture, has a strict timeline for submission of requests for resources, such as satellite access and data network Interim Approval To Operate (IATO). Errors in submission hamper this process and cause unnecessary delays.

3. Stakeholder High-level Goals

- Timely and accurate common operational picture of the battle space
- Reliable, robust and on-demand ability to communicate internally and externally
- Seamless ability to complete tasks that require the use of communications networks
- Accuracy in communication resource requests to external agencies

4. User Summary

System users can be divided into four categories:

- System Administrators:
  Responsible for maintaining the software system and administering user accounts and permissions.
- Network Planners:
  Those persons that are involved in planning the communications infrastructure.
- Plan Implementers:
  Those persons that use the network plan to install and maintain the infrastructure.
- Need To Know:
  Those persons that require knowledge of the infrastructure for future and/or adjacent planning and coordination.

5. User High-level Goals

- Common knowledge of the most current network plan, as well as previous versions of the network topology
- Asynchronous and synchronous collaboration between planners
- Better accuracy and efficiency in the planning process
- Easily create, modify and delete planning projects

6. User Environment

Users will be operating in both garrison and deployed environments. In both cases, access to the network server on which the application is installed is required for its use. Because of the uncertainty and variation in the speed and

reliability of deployed communication networks, the application must also be designed so that it minimizes bandwidth use without negatively affecting the application usability.

G. OBJECTIVES AND SUCCESS CRITERIA

The system should be designed to achieve the following objectives:

1. Indirect objectives:

- Enhance the common operational picture of the battle space by enabling a common knowledge of the communications network.
- Enable planning of networks that provide reliable, robust and on-demand ability to communicate internally and externally, as well as the seamless ability to complete tasks that require the use of communications networks.
- Accuracy in communication resource requests to external agencies.

2. Direct objectives:

- Consistent knowledge of the most current network plan, as well as previous versions of the network topology, in order to enable a common understanding of the network by all stakeholders.
- Centralized planning environment providing asynchronous and synchronous collaboration between communications planners.
- Enhanced accuracy and efficiency in the planning process.
- Ability to easily create, modify and delete communications planning projects.
- Minimize error inherent in the current manual process.
- Portability through the ability to install multiple versions on separate networks (e.g., NIPR and/or SIPR)
- Ability to efficiently and effectively administrate system.

H. USE CASES

Analysis of the system objectives presented in the previous section provides a basis on which to determine expected use-case scenarios for the CCP system. Based on this analysis, we identified eight key use cases that represent the functionality required to meet these system objectives. These use cases are depicted in Figure 3. (See Appendix B for descriptions of each Use Case.)

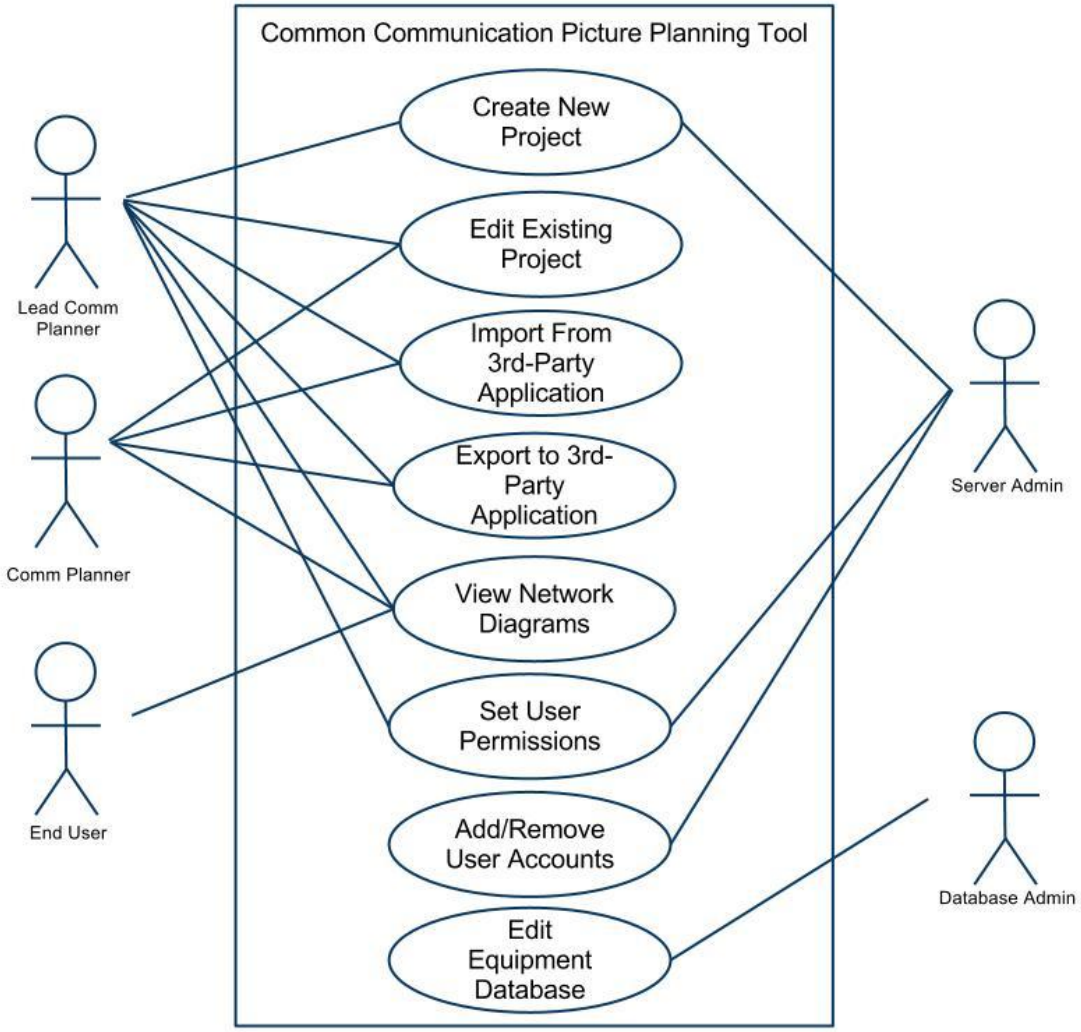


Figure 3. CCP use cases diagram

I. CCP SYSTEM DOMAIN MODEL

There are specific domain entities inherent in all communications networks which are included in all software designed to operate in the domain of communications network planning and monitoring. While their names may differ slightly from one product to the next, the underlying concepts do not.

A communications network can be broken down into five primary entities: Site, Link, Circuit, Equipment and User. All networks consist of a unique combination of Sites, Links, Circuits and Equipment. The domain model for the CCP system was designed with this understanding. Figure 4 shows a conceptual model that is robust enough to meet our desired goals, yet generic enough to enable ease of information sharing between any number of communications planning and monitoring systems.

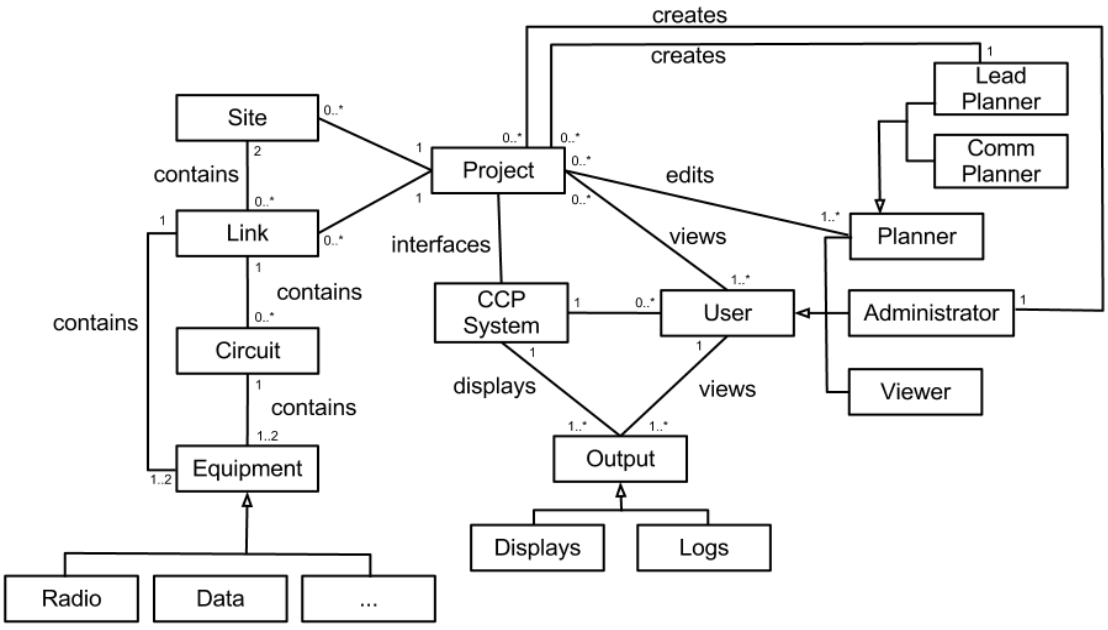


Figure 4. CCP communications planning system domain model

J. CONCLUSION

As previously stated, the goal of this thesis is to model a software system that enables a more efficient communications planning process by addressing the areas of automation, collaboration between planners and interoperability of software applications. We have identified the need for a system that automates the process of collaboratively creating communications network diagrams in order to reduce the time a planner currently spends creating these diagrams manually, with the added benefit of minimizing human error. The following two chapters discuss in more detail the design and implementation factors that should be taken into consideration in the software engineering process.

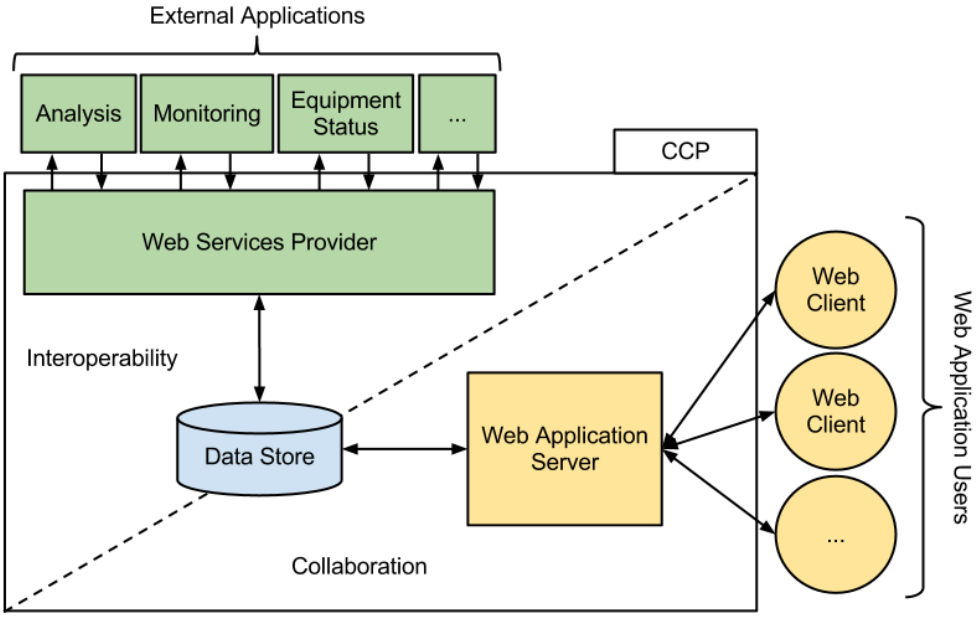


Figure 5. CCP collaboration and interoperability

Chapter III specifically addresses designing software for collaboration between communications planners in a cloud environment. It proposes software design patterns best suited for the context of cloud-based communications planning collaboration. Likewise, Chapter IV covers how a more efficient planning process can be achieved by removing some of the redundant manual work that

results from lack of interoperability between communications planning and monitoring software systems. It presents a method of implementation that enables interoperability between existing and future systems through identification and sharing of redundant communications planning and monitoring system entities and information sets.

As shown in Figure 5, CCP is designed to encapsulate these techniques in a single system. Chapter V provides an overview of the resultant proof-of-concept application.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SOFTWARE DESIGN CONSIDERATIONS FOR CLOUD-BASED COLLABORATION

A. INTRODUCTION

By leveraging current technology and software design techniques, coupled with the availability and prevalence of high-speed networks, it is possible to develop an application that enables a more efficient communications planning workflow. Part of this increased efficiency comes as a result of streamlining collaboration between planners, thus reducing or removing redundant activities in the planning workflow and resulting in a faster planning cycle. For cloud-based near real-time collaboration to occur in an operational environment, the application must be designed so that its speed is comparable to desktop-based solutions, yet is efficient in terms of both network usage and server load. This chapter describes the recommended design patterns and implementation techniques best suited to meet these criteria.

B. COLLABORATION AND COORDINATION REQUIREMENTS IN COMMUNICATIONS PLANNING

Planning communications networks is inherently a cooperative and collaborative process. Throughout the iterative planning cycle, there are multiple types of collaboration that must occur in order to ensure an implementable and feasible solution. The first, and most obvious, type of collaboration that occurs is between planners of the same unit. Working on their specific portion of the network, subject matter experts of different plan areas work together to ensure a network plan that is both cohesive and feasible. For instance, a single network link has the capability of containing multiple distinct circuits, for example: NIPR, SIPR, and voice. Because each circuit type is its own specialized type, each requires a planner knowledgeable in that field. In our example, two members of the SPE cell, the Data Officer and the Telephone Officer, would hold the responsibility of planning the NIPR and SIPR, and voice circuits, respectively. In order to plan properly, each planner must know details of the other's circuit (such

as the amount of bandwidth used). In a more complicated network, such as one where the voice circuits are routed over the data network (such as a VOIP circuit), the level of required coordination is dramatically increased. It is for this reason that the members of the SPE cell generally share a common office space.

The second type of collaboration that occurs is between planners from different units. This relationship is normally between senior and subordinate units, or supported and supporting units, but can occasionally occur between adjacent units. In a senior/subordinate relationship, the senior unit provides the subordinate unit with the information needed for the subordinate unit to plan its portion of the network. This information usually consists of equipment types, bandwidth, and allocation of IP addresses and telephone numbers. It is important to note that this coordination is not one-way. If the senior unit were to dictate the plan without coordination, the resulting plan could be flawed due to outdated information. The subordinate unit participates throughout the plans development phase by providing information on equipment and personnel availability and plan feasibility. Coordination between adjacent units is needed in troubleshooting a link between them. In the event of adjacent units, one of the units is designated as the link establishing agency, and one is designated the link terminating agency. The establishing agency assumes the responsibility for maintaining the link, and directs all troubleshooting actions.

The third type of collaboration is between locations. This is often the case once the network has been installed, and is being monitored by SYSCON and TECHCON watch personnel. Because communications networks in operational environments are dynamic in nature, with units regularly changing locations, as well as equipment changes and upgrades, it is important that units on both ends of all network links have a common knowledge of the network topology and configuration. In the event of a planned or unexpected change to the network plan, everyone on that particular network should be notified of the change. Once notified, each respective location must then review the updated network plan and make changes accordingly. The action of actually changing the network is

controlled by the senior SYSCON and TECHCON agency, and is outside the scope of this document. However, in order to facilitate the required changes, everyone involved must have a common knowledge of the post-change architecture. This common knowledge is provided through the information represented on network diagrams produced as an output of the planning process.

The common thread that ties the various types of coordination together is the necessity for all parties to maintain a constant and common knowledge of the current network plan. This can prove to be difficult in such a dynamic environment due to the pace and number of changes. When this common operating picture is lost, problems begin to manifest themselves. One of the simplest examples of what could happen is the use of outdated diagrams. At first glance, a simple solution could be to hold off on distributing the plan until it has been finalized, and only then provide the plan to the personnel responsible for actual installation and operation of the network. This approach has two problems: first, by not providing draft plans as they become available, you deny personnel at all levels the time needed to conduct the required adjacent planning and coordination needed for actual implementation, such as submitting requests for materials, finalizing any maintenance actions, and advising planners early if a portion of the plan is untenable. In this respect, draft network diagrams serve as a type of warning order. The second problem with not distributing draft plans is that even in the best situations, the plan often changes after it has been “finalized” due to unforeseen events. In essence, the understanding is that network plans, and the diagrams that depict them, are living documents that require constant tweaking. Providing draft copies as they become available serve a similar purpose as the Warning Order does in the operational planning cycle.

Because of the multi-layered, multi-threaded nature of collaboration in the communications planning process, a certain level of overhead in planning time is induced. By focusing on the reduction of duplicated efforts and the use of automation to remove human interaction where possible and feasible, the

planning process can be streamlined to a level of efficiency more suitable to the fast-paced nature of modern-day military operations.

C. CURRENT COLLABORATION TOOLS USED IN COMMUNICATIONS PLANNING

As operational planning is time constrained, organizations must employ techniques that reduce the overall time required to complete the planning process [11]. The goal of improving efficiency through the incorporation of emerging technology as it becomes available is not new to the Marine Corps. The widespread availability of personal computers resulted in a great leap forward in planning efficiency by eliminating the inefficiencies of creating and updating hand-drawn diagrams. Development and implementation of computer networks resulted in the ability to distribute planning documents more efficiently in terms of both time and space. Both of these advances helped to speed up the planning process, at the same time enabling a better-coordinated plan. Many of the techniques that were adopted in the last two decades to improve planning efficiency are still in use today. The use of these techniques has improved the flow of information, both within and among Marine Corps organizations and support improved situational awareness and collaborative planning. The use of networking has also provided the ability for electronic reach-back, which can result in reduced deployed staff size through the use of geographically distant personnel assets [12].

Due of the continually changing nature of the network plan, several tools are used to facilitate the required collaboration and sharing of information between planners. The three most common tools used by Marine Corps planners today are e-mail, shared network folders, and server-based document management tools such as Microsoft SharePoint [13]. Each provides the ability for planners to collaborate asynchronously throughout the planning process, have undoubtedly improved the ability of planners to collaborate, and in turn have enhanced the overall efficiency of the planning process. Each provides distinct advantages and disadvantages.

Collaboration via e-mail is the most basic, most fragmented and most used of the three techniques. It is also the method most likely to result in duplication of effort and version control errors. In a typical scenario, planners work on their portion of the network plan individually, and e-mail their updated versions to either a central planner (who maintains the “master” copy on which all changes are applied), or to all other planners. In the context of communications planning, this process is the least desirable of the collaboration methods. It succeeds in allowing multiple planners to work simultaneously. However, it is inefficient in terms of both workflow and resources. It also requires manual version control and discrepancy detection.

The next level of collaboration is the use of network-based shared folders. Providing all involved planners access to a shared network resource resolves some of the inefficiency inherent in e-mail-based collaboration. In particular, each planner’s portion of the plan can be saved in a central location, and is viewable by everyone with access to the shared folder. This method still requires a designated person to combine all of the plan subsections into a master document, and to ensure that the most up-to-date version of each section is represented. Instead of each planner maintaining and updating a separate file for his portion of the plan, another method used is for all planners to edit the same document. This can be accomplished by either creating a local copy of the document, making changes, and replacing the shared document with the edited version, or directly editing the document (the problem with editing local copies is discussed further below). If directly editing the shared copy, only one planner at a time has the ability to make any changes, with all other planners having read-only access to the file. The inability for multiple planners to work on the plan simultaneously is inefficient, making this technique less desirable.

Improving on the shared network folder technique, file management software such as Microsoft’s web-based SharePoint formalizes the concept of version control and centralized storage, reduces the number of required e-mails and the overall duplication of effort. Built in version control helps to alleviate

situations where multiple versions of the plan are in distribution. It also prevents planners from editing the document directly from the shared resource, thereby avoiding the inefficient single-threaded workflow seen above. As the source document cannot be edited directly, there is still a requirement for the planner to download a local copy of the file in order to make any changes.

Network folders and SharePoint both share a problem with simultaneous editing manifested by requiring planners to edit files locally. Take, for example, the following scenario: two planners need to make changes to their respective portions of the plan. The changes being made do not require the planners to coordinate with each other because they do not affect one another directly. Both planners download local copies of the most current version of the plan from the SharePoint server, make their respective changes, and then each uploads their edited version. The version uploaded last will be represented on the server as the most current. However, any changes made by the first planner will have been overwritten, as seen in Figure 6.

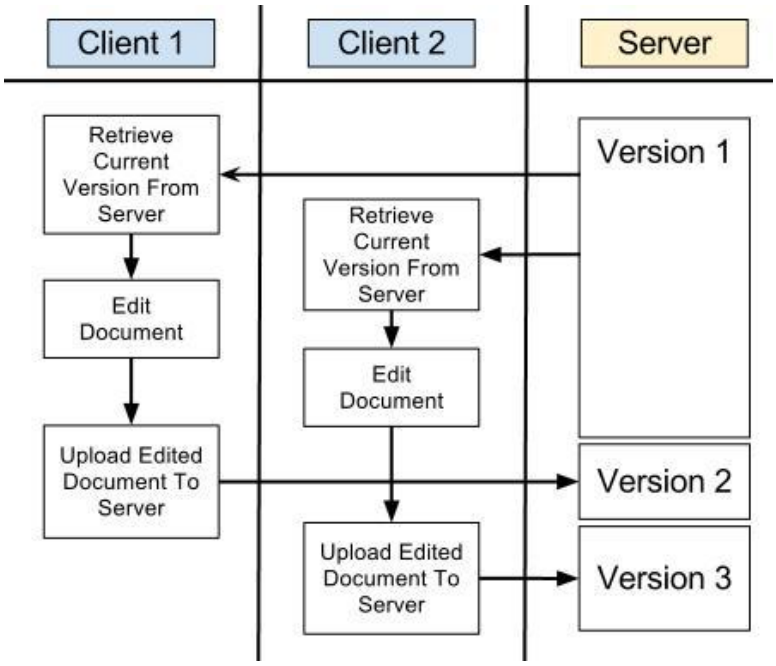


Figure 6. Version control and update discrepancy

Though not enabled by default, SharePoint attempts to alleviate this problem through the use of “pessimistic” concurrency control by introducing the notion of “checking out” documents for editing. Documents that are being edited are marked as “checked out” on the server, and other users are not able to edit the document until it has been “checked in” again. Also, users that wish to view the file will not see any changes made until it has been checked-in. While effective at preventing the situation described above, it creates the same inefficient work flow that we saw with network folders because the document can only be edited by a single user at a time, and those edits are not visible until the planner has finished. It also creates a new problem scenario where-in the planner that has checked out a document forgets to check it back in the repository for an extended period of time, thereby inadvertently denying other planners the ability to do any kind of work on the document.

A more ideal planning software application takes the next logical step in enabling collaboration. By allowing multiple planners the ability to edit different portions of a document simultaneously, with updates visible to all connected users in near real time, the inefficiencies of the “check-out” process are eliminated. One way in which this can be accomplished is through the use of a cloud-based solution. All users not only see changes in near real time, but are actually collaborating on the same instance of the document residing on the server.

As opposed to using strictly pessimistic locking, preventing multiple planners from working on the project at the same time, or optimistic concurrency control, wherein the most recent change is always applied, a better design choice for cloud-based systems is a combination of the two. This method allows two planners to make changes simultaneously, and in the event of a conflict the most recent change takes precedence. While this presents a problem similar to the versioning problem discussed above (see Figure 6), since each client is pushed updates in near real-time as they occur, the level of granularity is much finer. Instead of a planner possibly losing all changes made since he began the editing

process, only the most recent change has the possibility of being overwritten. Even in this instance, it would require that two planners happen to update the same portion of the document by calling the same update method at exactly the same time. This possibility can be prevented through the use of functionality provided in modern programming languages. For example, in Java, adding the "synchronized" keyword to a class method allows only one thread on the server to execute the method, blocking all other threads from executing that same method until completion [14]. Calls to other methods are still possible, and may occur in other threads.

D. SOFTWARE DESIGN PATTERNS FOR CLOUD-BASED COMMUNICATIONS PLANNING

While cloud-based solutions are similar to standard desktop applications in many ways, enabling simultaneous editing, collaboration and near real-time updating for multiple users presents many new challenges that must be addressed in the design of the software. There are certain structural design patterns that can be used in their standard form regardless of whether the application is desktop-based or web-based. For example, the Factory pattern could be used for the creation of any required graphical forms [15]. There is no discernible difference between the implementation requirements of the Factory pattern for either application platform. However, there are some patterns that are more appropriate choices than others for cloud-based applications. Furthermore, due to technological limitations and constraints, some design patterns require extension or modification in order to optimize their use and operate in a more efficient manner when designing for the cloud.

1. The Proxy Pattern

Without the use of a proxy, the process of updating the information for all attached clients each time a Project (see Appendix C) is updated by any of the clients is highly inefficient. The client proxy object, which is the remote proxy, is a remote representation of the object residing on the server, and implements the

same interface as the actual object on the server. When a client makes changes to the project, the remote proxy is invoked as if it were the actual object residing on the client. The remote proxy handles the task of communicating with the server, invoking the required remote method, and returning any results. In the case of a project change, the remote proxy is used to facilitate the process of updating the server, which in turn updates any underlying data store (database, XML file, etc.) and all other connected clients (Figure 7).

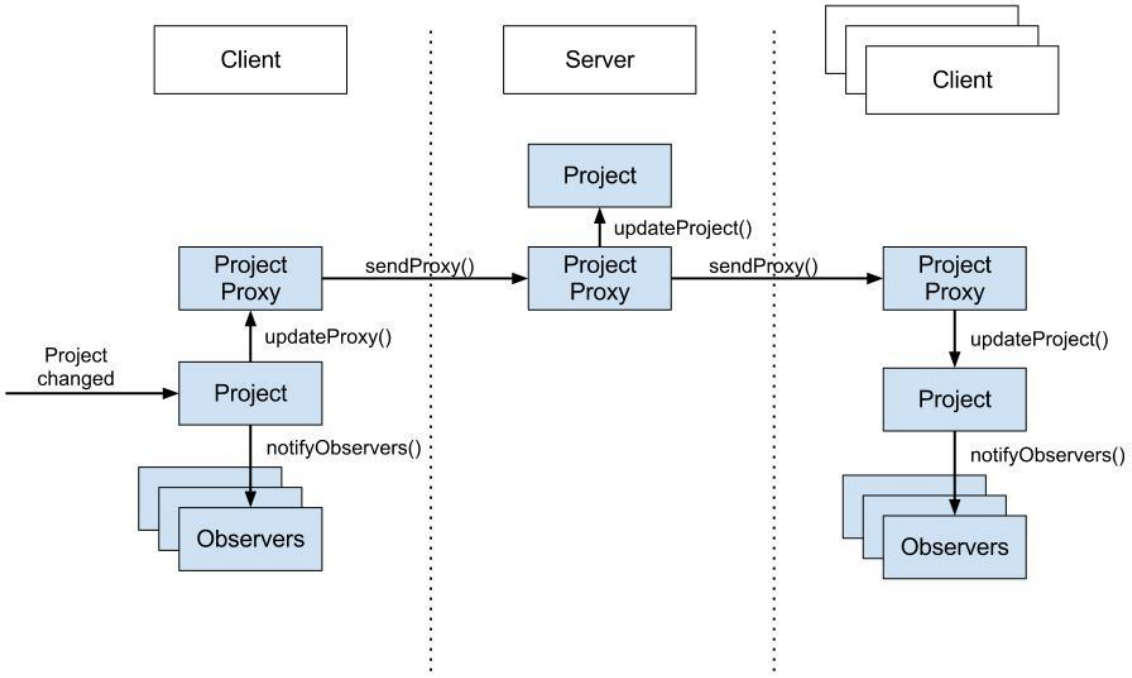


Figure 7. Passing Project using the proxy pattern

Used in this manner, the remote proxy provides a clean and seamless method of accessing and referencing objects and functionality on the server through the client interface. Accessing and updating remote objects could be accomplished without using a remote proxy, but doing so would result in a more cumbersome and less flexible design.

In order to take full advantage of the efficient communication mechanism that the remote proxy design enables, it is necessary to minimize the amount of

information transmitted between the client and the server. As an example, consider the use case of a client updating the project. Propagating the changes to the server, and to all other clients, can be accomplished using several techniques, each with its own advantages and disadvantages.

The first technique that can be used to minimize transmission size is to create specialized methods for each possible change. Using this technique, several methods must be written to account for all possible changes. When a change occurs on the client, the respective method is triggered, initiating an update on the server, which is then propagated to all other clients. This method necessitates the programming of complex logic in both client and server-side code that must be updated in the event of any change to the application structure by the software designers. Due to the number of methods that would be required in a non-trivial application, the result would be a large amount of specialized code, reducing re-usability and dramatically increasing testing and maintenance requirements. While understanding the implementation is fairly simple, the design, programming and testing processes can be very time-consuming, and do not guarantee that the software designer has captured all possible change scenarios.

The second technique simplifies the update process by removing the requirement for writing complex logic. This is accomplished by simply distributing the updated object each time the Project is modified. There are at least two problems with this technique in our particular scenario. First, it is highly inefficient. Even a minor alteration, such as a name change, results in the entire object being transmitted, including all of the information that has not been altered. In a cloud environment, minimizing the required bandwidth of the application has a direct effect on throughput speed and server load, each of which directly affects the user experience. Sending the entire object every time a change is made results in unnecessary bandwidth usage because of the transmission of superfluous data. Secondly, because the application relies on a graphical user interface (GUI), each time the project is modified, the GUI must

also be updated accordingly. In order to implement this functionality efficiently, the observer pattern is implemented, such that the Project is observed by the GUI items, and notifies them of any changes (Figure 8). The Project object extends the abstract IsObservable class, and maintains a collection of associated observers. By passing around the entire Project object between the server and clients, a problem manifests itself when the project needs to be transmitted to the server. Since the project maintains a list of local observers (GUI elements), and the entire object is being transmitted, this observer list will also be sent as part of the Project object. Other clients will receive the updated Project object from the server, with the Project's collection of observers pointing to the GUI objects from the originating client. In addition to inaccurate pointers, sending the entire Project with all observers incurs additional overhead. Passing entire objects around as opposed to using specialized methods makes for simpler code and easier testing. Unfortunately, the lack of efficiency counters these benefits. To avoid both of these problems, all subscribed observers can be removed from the Project before being sent, requiring all other clients to re-attach their list of local observers to the updated Project object each time an updated Project object is received.

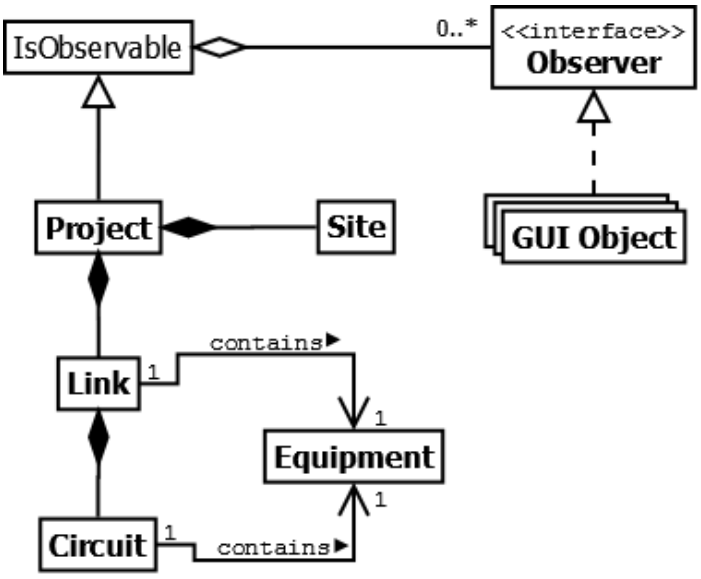


Figure 8. The Project object

2. The Data Transfer Object Pattern

The use of a proxy to marshal information between client and server provides the benefit of an efficient communication technique that we would like to take advantage of to generate code that is easy to understand, test, maintain and reuse. As discussed, it is more difficult to achieve these benefits using the multiple specialized methods technique. We can achieve these benefits by passing around the entire object, but we incur additional overhead and unnecessarily waste transmission time and resources. The inefficiency inherent in the latter technique can be partially resolved through the use of the Data Transfer Object (DTO) pattern. In the DTO pattern, a helper class known as a Data Transfer Object (DTO) provides a streamlined capsule for transmitting an object's critical information without needing to transfer the entire object. A secondary benefit of the DTO technique is that it also alleviates other problems that manifest themselves when passing around objects, such as the need to serialize multiple class objects and the need for continuously attaching and detaching observers from the Project object.

In its most basic form, the DTO object serves as a simple means of marshalling data between server and client, and vice-versa. The DTO contains the same fields as the object to be updated. Additionally, it has the respective "getter" and "setter" methods for updating these values. However, the DTO is void of any logic present in the object it represents, resulting in a smaller file size, meaning less bits that need to be transmitted. In order to accomplish this, there must be additional logic in the application to create and update the DTO. Use of the DTO pattern results in a degree of code redundancy because there are effectively two classes representing the same data. Whether the efficiency gained by using a DTO negates the redundancy is a consideration that the designer must take into account when deciding whether to use the DTO pattern.

In the context of our application, the benefits provided by the use of the DTO pattern in terms of reducing network and server load outweigh the concerns regarding redundant code. Moreover, a slight modification to the DTO pattern

can result in even more efficiency while requiring only slightly more logic. Similar to the way that the DTO is more efficient due to the smaller size as compared to the entire object, use of a partial DTO is more efficient than sending an entire DTO object. Used in this manner, the DTO provides nearly the simplicity of the second technique, coupled with the time and space efficiency of the first, without the extraneous code bloat. The concept driving the use of a partial DTO is that even the DTO carries extra information that can be trimmed. By using the DTO object as a shell to transmit only the fields that are relevant to a particular update, we further increase efficiency (see Appendix D).

To demonstrate the efficiency of using a partial DTO, let us examine the case where-in one of the application user clients makes a single change to the Project, such as changing an object's position on the diagram. In the least efficient case, the entire Project object, including its lists of Site and Link objects and all of their respective elements (Figure 8), is sent to the server, who then sends it to all connected clients. Each client must update its Project object to match the newly received one (which can be accomplished by pointing to the new object). A more efficient technique is to send the Project's DTO instead of the whole object (Figure 9). Now, instead of sending the entire object, only the Project's information is encapsulated in a DTO, which is then transmitted to the server and distributed to all other clients. However, there is still a level of unnecessary inefficiency, because we are only concerned with the changes that were made, and the DTO contains all of the project's information. While all of the unnecessary code has been stripped off in the DTO, it still has all of its fields populated. In order to send only changes, the Project DTO can still be used, but this time only as a shell. Instead of populating all fields of the DTO when the Project is modified, only those fields that have changed are populated. All other fields are left null. On receipt of the DTO, the server updates its stored Project object to reflect the new values contained in the DTO, and in turn sends all

attached clients the minimal DTO object so that they can update their local Project objects in the same manner. The result is that only the changes have been transmitted.

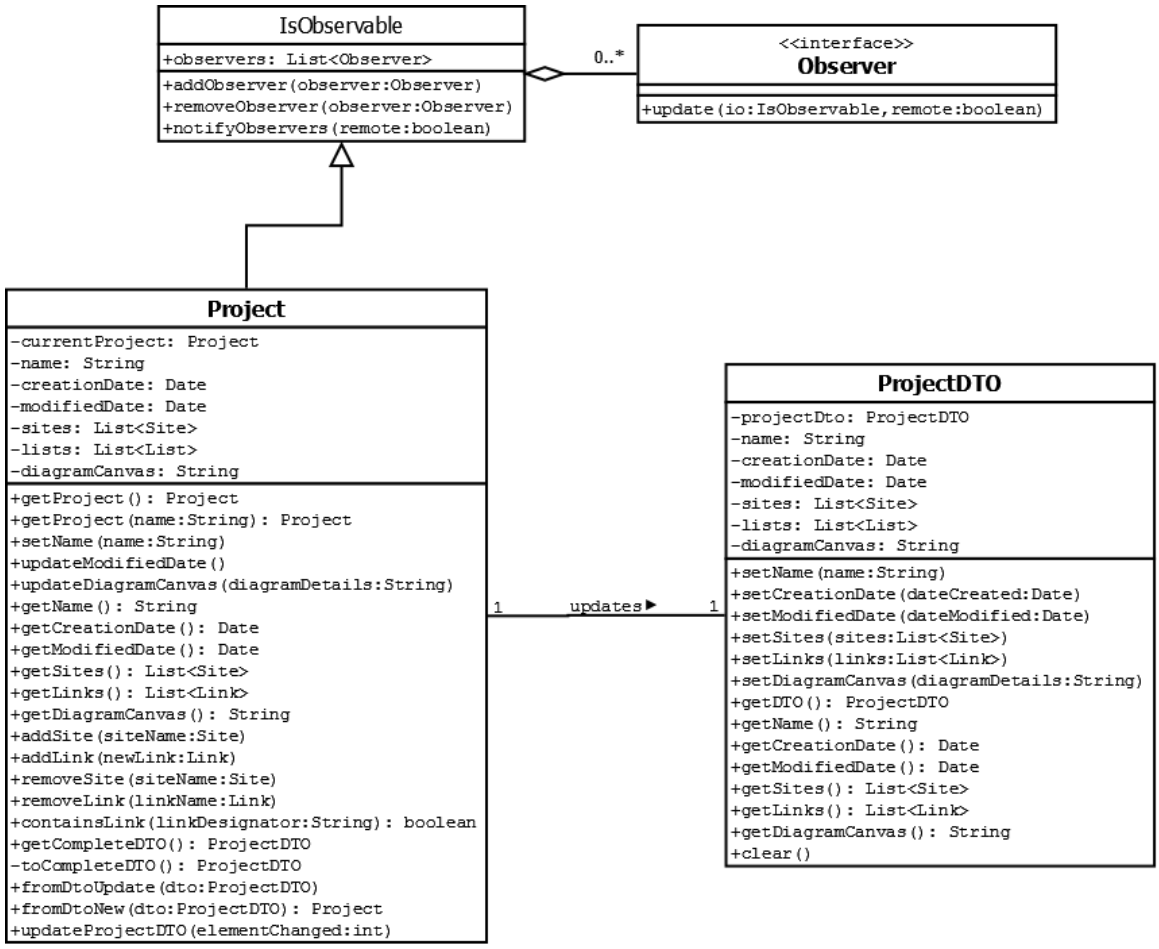


Figure 9. ProjectDTO

3. The Observer Manager Pattern

The Observer pattern (also known as publisher/subscriber) is used extensively in many GUI-oriented designs as a means of updating GUI elements in response to changes to the underlying data. This minimizes the level of coupling required. Each GUI element that needs to be updated registers itself with the object of concern. This object, having extended the observable class (which we have named IsObservable in our application), maintains a list of

observers, and contains a method to notify all observers in the event of a change. Once notified, the observer's update method is triggered (Figure 10).

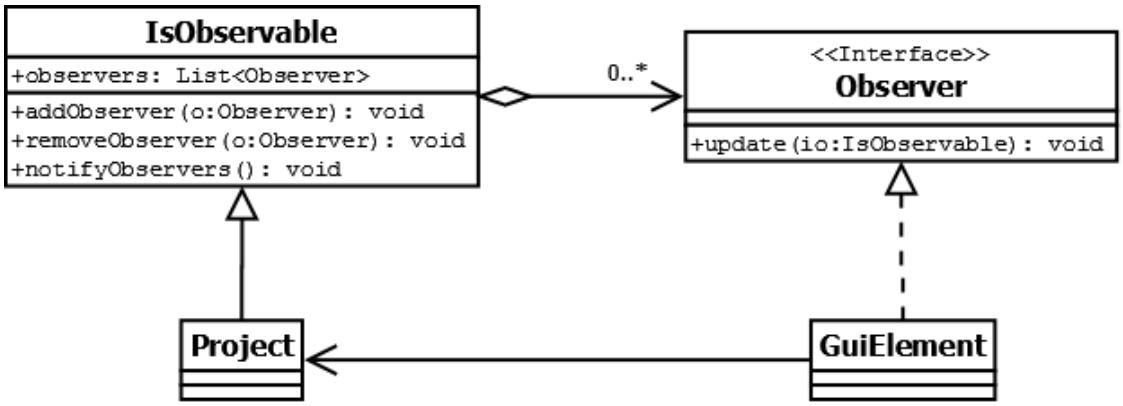


Figure 10. Applying the traditional Observer pattern

The traditional Observer pattern works well in situations wherein the object that serves as the source of the underlying data does not change. However, if the source of the data has the ability to change there is no clean method of updating all observers so that they reference the new object. For example, in our application, there exists a scenario in which the planner could decide to work on a different project. At the point where the application initializes the new Project object, all GUI elements still reflect the information from the previous Project, and are still registered observers of the old Project object. The task of populating the new Project's collection of observers is tedious, requiring all objects to re-register as observers of the new object. This problem can be alleviated through the creation of a class that specializes in the management of observer relationships (Figure 11).

By using an expert class to control observer relationships, we create a more loosely coupled observer structure, allowing the project that is being worked on by the user to be changed without having to update all other project elements that are observing it. As shown in Figure 12, the managing class maintains a mapping of observers to observables. Updating the GUI to reflect a

new or different project only requires a remapping of observers to the new observed expert object at run-time, without having to modify any observer or observable code or introduce complex conditional statements and redundant code throughout the application.

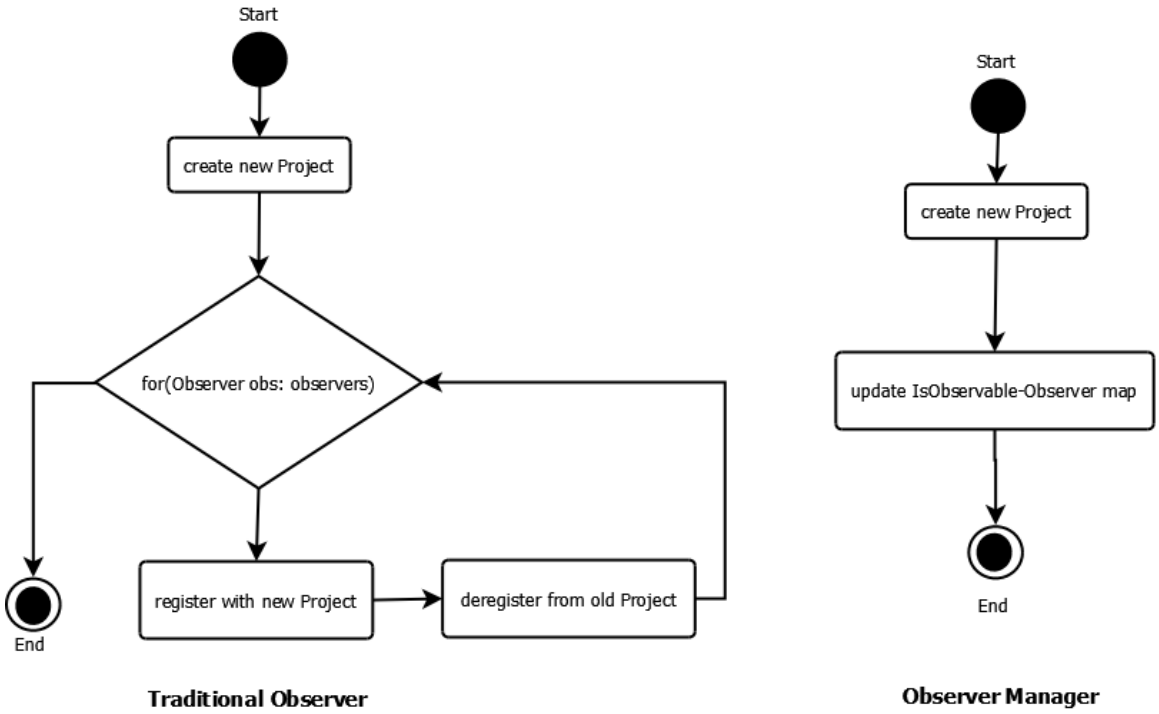


Figure 11. Comparison of adding observers to new project

The expert class in this case is called the Observer Manager (see Appendix E). The manager does not alter the well-established and defined traditional Observer pattern. Instead, it extends it and adds a layer of abstraction that results in more modular code. A key feature of the manager is that it relies on multiple inheritance, extending the IsObservable class (see Appendix F) and also implementing the Observer interface[1] (see Appendix G). An object that has implemented the Observable interface registers itself with the manager via the manager's addObserver() method, passing in the IsObservable object that it

[1] Because the manager is also an IsObservable object, it is technically possible for it to "manage" itself. For our application, this behavior is not needed or desirable, so all references to IsObservable objects being mapped by the manager explicitly exclude the manager itself.

wishes to observe as a parameter. The new observer object is added to the list of objects that are observing a given IsObservable, maintained as a map by the observer manager. If the manager does not already have a map entry for the IsObservable object, the manager creates the entry and also registers itself as an observer of the object. When an IsObservable object other than the manager is modified, instead of triggering the update of all its observers, it merely notifies the manager, which in turn calls the update method of the observers that are mapped to the object that triggered the update.

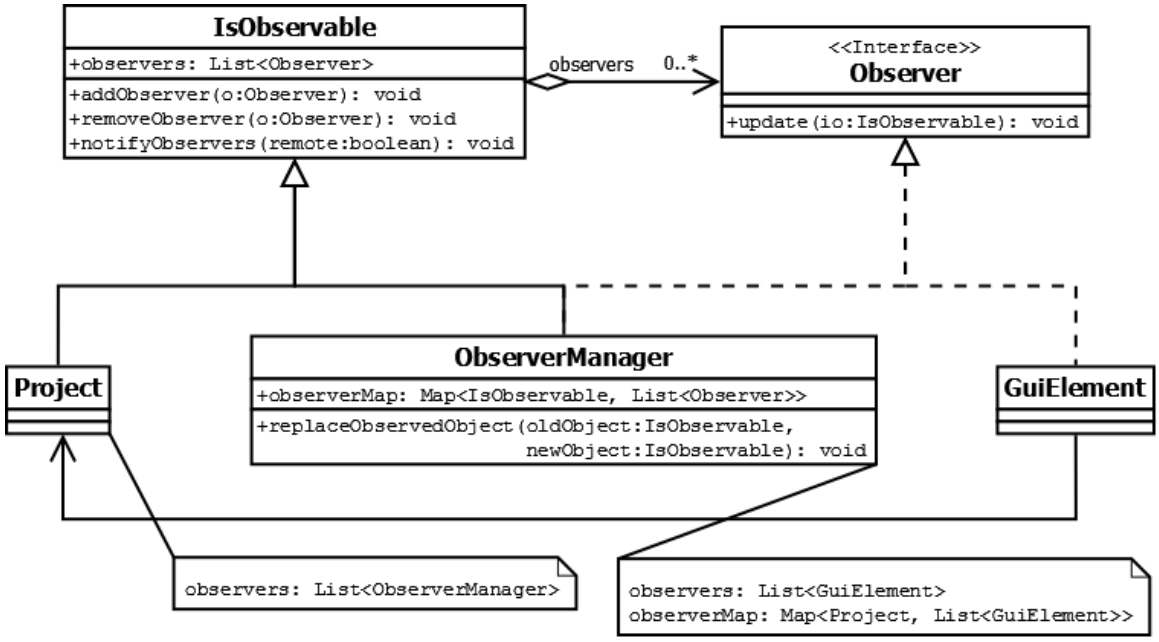


Figure 12. Observer Manager class diagram

Additionally, the observer manager contains a special method called replaceObservedObject(). This method can be called whenever an IsObservable object is replaced by a new object. Instead of requiring all observers to re-register with the new object, the observer manager simply updates the IsObservable-Observer map to reflect the change. Since the observers are observing the manager and not directly observing the desired IsObservable object, the swap is simple and instantaneous, requiring no further code.

The final required addition to the Observer pattern is the use of a Boolean flag in the notifyObservers() method to indicate whether the update was triggered from a change on the server, or from a change by the user on the local application. This is important because one of the goals of our application design is to structure the application in a manner that facilitates collaboration amongst planners working simultaneously on the same project.

The benefit of using this update source flag can be easily demonstrated with a simple example. In our application, updating observers occurs in two dimensions: sending and receiving updates to and from the server, and updating the local display. Without a flag indicating the update source, the application would enter an endless loop of updating.

4. Architectural Software Design Patterns

There are many architectural considerations that need to be made when designing collaborative software that relies on near real-time updates for all users. Several methods exist that provide this capability, such as Java's Remote Method Invocation (RMI). Many of these products make use of the Transmission Control Protocol/Internet Protocol (TCP/IP) for establishing and maintaining a persistent connection between the client and server in the form of a socket. By establishing a socket between the client desktop application and the server, each is able to invoke methods on the other, providing the capability to "push" updates in real time as they occur.

In designing a cloud-based application, one of the goals is that the application is accessible from any computer, without the requirement to download or install additional software. This allows the planner to access the application and collaborate with other planners from any location, and from almost any device. It also allows the application to run on networks that block certain types of TCP/IP traffic without having to add additional access control rules. This can be accomplished on modern-day computers through the use of the Internet as a network backbone, using the web browser as the user interface.

Today's web browsers are capable of providing robust applications that rival their desktop counterparts in functionality. Through the use of Asynchronous JavaScript and XML (AJAX) development techniques, dynamic applications can be designed to perform much of the program's logic on the client machine, reducing overall server load and accomplishing one of our design goals.

One of the key distinctions between using a browser-based application and a standard desktop application is not one of processing power, but of protocol limitations. By their definition, web browsers are designed to implement the Hypertext Transfer Protocol (HTTP). This presents a problem that directly affects the architectural design decisions that need to be made.

Unlike TCP/IP, HTTP does not currently provide the ability to maintain a persistent communications socket between clients and servers. At the time of this writing, the World Wide Web Consortium (W3C) is in the process of developing standards for a new protocol known as WebSocket, which will provide the ability to establish TCP sockets, and thus real-time communication between client and server, through web browsers [16]. However, no modern browsers officially support this functionality, and the standard has yet to be finalized. Thus, in order to take advantage of the benefits provided by a web-browser based application, our application is bound by the restrictions imposed on it by the current HTTP protocol.

The key restriction that will affect our design decisions is that HTTP only supports a server sending responses to client requests. This means that the server cannot "push" information to the client in real time without some type of client invocation. This is not a problem for most browser applications, because most do not require real-time updating. A good example of this that is common today is a news aggregator. Several websites provide the ability to retrieve and display new stories from sources of the user's choice. The news feeds are updated periodically based on the client application settings. The lack of an ability for the server to update the news feeds in real time as the source changes is inconsequential. If the instance arises where the user wants to get the latest

articles and the application has not updated recently, the user can usually click a button to initiate an update. Through the use of AJAX, the client application makes an asynchronous HTTP request to the server, which executes the request and responds to the client with updates.

Before proceeding further, it is necessary to discuss the importance of this asynchronous call to the server because it highlights another factor that affects design considerations. The JavaScript language on which we rely to provide client-side code execution is not multi-threaded. If the server call were synchronous, the application would freeze until the server completed its task and returned a response. This type of event is referred to as “blocking” because it prevents any other event from executing until it is complete. By making an asynchronous call, the event is no longer blocking. The client sends the HTTP request to the server and executes the next programmed event without waiting for the server to respond. The server maintains a reference to the requesting client and returns a response when it is ready. The client acts on the response when it is received. As mentioned in the previous paragraph, server-initiated communication to a browser-based client is not supported by the current HTTP protocol; the browser-based client must periodically send requests to the server for updates.

The process of periodically checking the server for updates is known as polling. By specifying a polling interval, the software designer can manipulate how often the client is updated. The length of the polling interval is directly proportional to the amount of latency in updating. Any changes that are made on the server will not be manifested on the client until the client polls the server for an update. We can implement a design that approximates real-time updates by merely reducing the polling interval to a point that the updates appear to the client in near real-time, such as every second. This technique is simple to implement and understand. However, it presents two major concerns. First, it

wastes a lot of network resources by requiring every client to continuously poll the server every second regardless of whether the server has been updated. Second, it strains server resources.

While this may not be a concern for network applications where the user base is small, it does not scale well for applications with a large user base. The following example illustrates the problem. Assume a web-based application that simulates near real-time updates by polling the server once per second is accessed by 1000 users. If all users were using the application at the same time, they would generate 1000 server requests a second. If each request were 128 bytes in size, the network bandwidth usage would total nearly 1 megabit per second (Mbps):

128 Bytes per user = 1024 bits per user

1024 bits per user x 1000 users = 0.976 megabits

This is significant considering the fact that many of the tactical networks that are in use today in operational environments have a total bandwidth of 1–2 Mbps. It would be infeasible and ill-advised to host the application server on a tactical network with this level of network overhead. Doing so could have an obvious negative impact on other command and control systems relying on the same network.

Since simple polling as described above is so inefficient, we must examine other methods for updating the client. Ideally, the client and server would only communicate when needed, either by the client sending updates to the server, or the server pushing updates to the clients. Desktop applications that use this functionality typically implement the observer pattern such that the client application observes the server, which notifies the client of any changes through the client's update() method. Until a protocol is agreed-upon and finalized that provides web browsers with a full-duplex socket implementation (such as WebSocket), this is not an option for browser-based applications. Fortunately, we can apply some specific techniques that take advantage of the properties of

HTTP in order to closely approximate a socket using polling, while minimizing the network overhead to a manageable, if not unnoticeable, level. The result is a similar observer pattern implementation.

One technique that can be used to implement polling in a more efficient manner relies on the use of a simple algorithm that adjusts the client's polling interval based on client and/or server activity. One popular example was written by Neil Fraser of Google, and is implemented by the Google MobWrite application [17]. Here is a sample Java implementation of the algorithm:

```
protected void computeSyncInterval() {
      int range = maxSyncInterval - minSyncInterval;
      if (clientChanged) {
            // Client-side activity.
            // Cut the sync interval by 40% of the min-max range.
            syncInterval -= range * 0.4;
            clientChanged = false;
      }
      if (serverChanged) {
            // Server-side activity.
            // Cut the sync interval by 20% of the min-max range.
            syncInterval -= range * 0.2;
            serverChanged = false;
      }
      if (!clientChanged && !serverChanged) {
            // No activity.
            // Let interval grow by 10% of the min-max range.
            syncInterval += range * 0.1;
      }
      // Keep the sync interval constrained between min and max.
      syncInterval = Math.max(minSyncInterval, syncInterval);
      syncInterval = Math.min(maxSyncInterval, syncInterval);
}
```

The algorithm works by incrementally increasing and decreasing the polling interval between the preset minimum and maximum time intervals. This technique provides a significant savings in bandwidth when little to no updates are made. Setting a maximum polling interval of ten seconds (as opposed to the one second interval used in the example above) results in a 90% decrease in unnecessary network usage.

The adaptive polling method is more efficient than simple polling, but since we've increased the maximum polling interval, it now suffers from the same problem we saw in Figure 6, albeit on a smaller scale. If an update has not been made in a while, and the polling interval is at or near the predefined maximum, we increase the possibility of conflict if more than one user decides to make an update within the same update interval. The last change made before the next polling event would be propagated to all other users, who would lose any changes that they may have made during the same interval.

In order to effectively implement the observer pattern to enable near real-time collaboration within the confines of polling, the time between the server being updated and the client being notified must be as small as technically possible. We can accomplish this by taking advantage of the features of the HTTP protocol through a technique called "long-polling." Using HTTP, the server can only respond to a request sent by a client. However, once a request is received the server maintains the reference to the client until it has sent its response. We can take advantage of this property by delaying the server response until an update is available. If the client sends a request and there are updates available on the server, the server responds immediately. However, if there are no updates available, the application can suspend the thread on the server for a set period of time (which would in turn put the client on hold if we were not using AJAX). If the server is updated or the time period expires, the thread is unsuspended and the appropriate response is sent to the client. On receipt of the server response, the client acts on the response appropriately by making any required updates and sends another update request to the server. By

always maintaining an open connection with the server, the time between the server receiving an update and all connected clients being notified is reduced to the speed of the network.

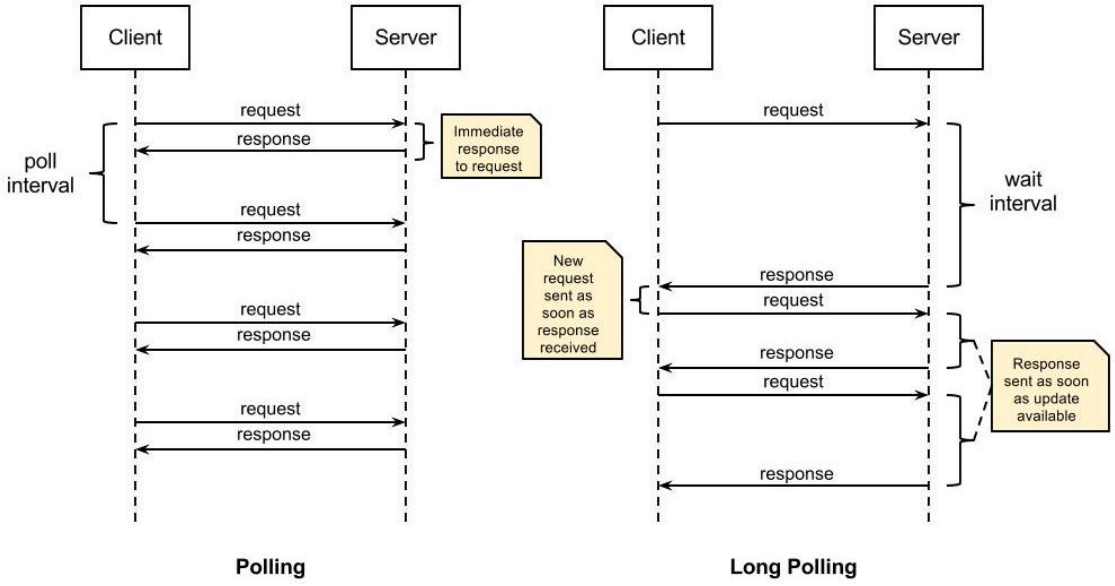


Figure 13. Polling versus long polling

The use of long-polling allows us to achieve near real-time update capability, while still enjoying the benefits afforded to us through the use of an HTTP-based web application. Of course, the choice of implementation is based on many factors. One factor that may affect the choice to use long-polling is a result of another technical constraint. The server requires one thread per waiting client. In a normal HTTP application, this is not a problem because the length of time for one client request is so small. Thus, in a larger application where a large number of users can be expected to access the application simultaneously, the use of long-polling requires changes to the server configuration to adjust the number of possible threads. Additionally, many server software installations now provide options that, when enabled, modify their thread-handling methods in order to provide scalability for applications that have large user bases and use techniques such as long-polling [18].

Further efficiency gains were realized via updates to the HTTP protocol. Through version 1.0, the HTTP protocol allowed only a single request/response pair per connection [19]. Version 1.1 introduced the concept of persistent connections, allowing multiple requests and responses over a single connection in order to the reduce latency incurred by having to renegotiate TCP connections.

E. LOGICAL ARCHITECTURE

In designing the logical architecture of the proposed system, the overarching goal is to improve the communications planning and collaboration process, thus enhancing and improving the ability of communications personnel to enable command and control. As discussed previously, this goal is achievable by taking advantage of the benefits provided by developing a cloud-based system.

The proposed system uses the Model-View-Controller (MVC) software design pattern, which was developed in the late 1970s for the Smalltalk platform and has since been included as an integral part of most modern user interface frameworks [20]. MVC frameworks are built around three main components: the Model provides an interface with the database, the View provides the graphical presentation to the system user, and the Controller establishes and enforces the rules and actions that take place in response to user input. Decoupling the system's roles into these three distinct objects as shown in Figure 14 allows us to design a system that is more easily developed, tested, updated and maintained.

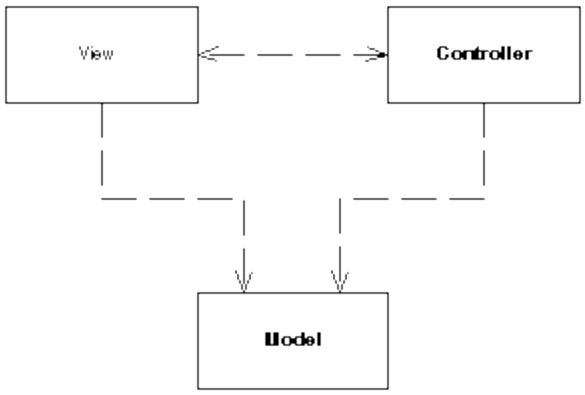


Figure 14. Generic model-view-controller logical architecture. From [20].

1. Model

The model of the proposed system consists of two parts: a client model and a server model (Figure 15). The server model is the Project object maintained on the server. It also provides an interface to the server data store (e.g., a relational database). Upon connection to the server, each client references the server's instantiation of the Project object representing the desired project. If the desired Project is not instantiated (because it is not currently being worked on by another system user), the server retrieves the Project information from the data store and instantiates a new Project. All clients working on a specific project reference and update the same Project object.

A change made to the Project object (the model) by one of the clients is transmitted to the server, which persists the change in the data store, and propagates it to all other connected clients. On the client, the model data is represented as a Project object, which is viewed and manipulated by the system user in the creation of a network plan. An important feature of the model is that it requires no knowledge of the view or controller. This allows the information stored in the model to be represented in different ways by any number of views, presenting only the information relevant to a particular context. For example, in the CCP system, the view presented to the planner differs based on which portion of the plan he is working on. A view of the transmission diagram will be

different from a view of the multiplexing diagram. However, they will both reference the same model. Likewise, two planners working on the same project simultaneously are manipulating the same model data on two physically separate views.

2. View

The View presents an interface with which the application user can view and edit data contained in the model. The client view is linked to the client model by implementation of the Observer Manager class presented in section D3. Changes to the model result in changes to the GUI elements that are observing the model data in a specific view. For example, if the client application receives an update from the server indicating the a new Site has been added to the Project, updating the local Project object to reflect the changes triggers the update() method of all observing GUI elements.

3. Controller

When a user interacts with a view, the controller is responsible for determining what should be done in response to user input. In most cases, this input results in changes to model data. The controller translates these interactions by providing the logic involved in validating the request and applying all requested change to the model. In essence, the controller serves as the “middle man” between the view and the model.

In the proposed system, the controller has two parts: the client side and the server side. All interaction between the client and server occurs in the scope of the client controller and server controller, which communicate via the Proxy class discussed in section D1. The client side of the controller is responsible for validating user input and making appropriate changes to the model (the Project object). As stated before, these changes trigger the update method of the Observer Manager, which in turn triggers updates of all registered GUI elements. In addition to updating the local model, the client controller is also responsible for sending these updates, packaged in a partial DTO, to the server controller via

remote proxy, and receiving updates from the remote proxy to update the client's local model data. Like the View, the Controller observes the model via the Observer Manager and responds to model changes through Observer update() methods. The server controller is responsible for ensuring that the server model is always up to date, and that all clients are notified appropriately.

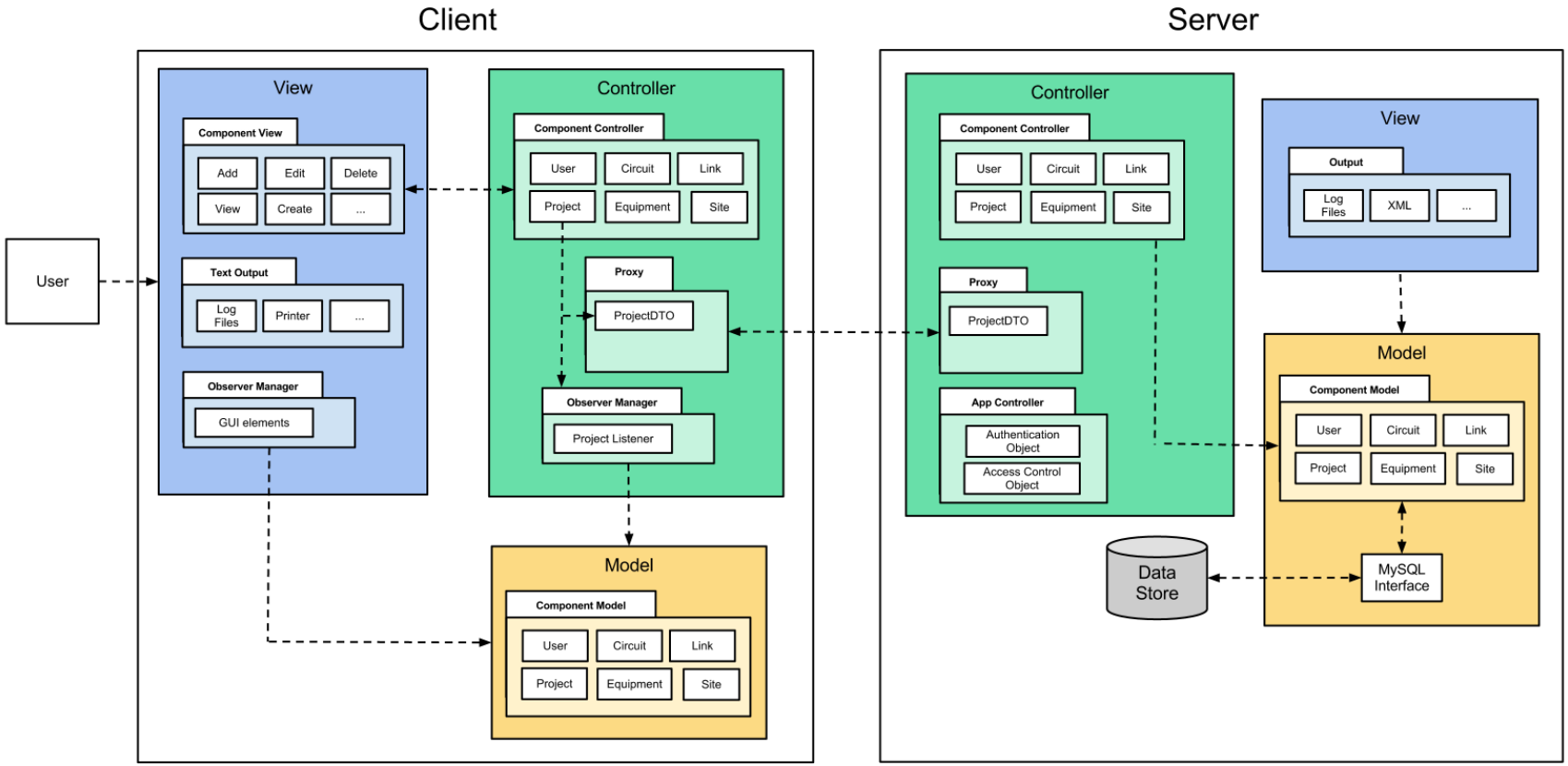


Figure 15. CCP model-view-controller architecture

F. CONCLUSION

The choice of appropriate software design patterns and logical architecture has a direct impact on the ability of planners to collaborate in a cloud-based environment. Leveraging the benefits of a web-based application by extending time-tested software designs in order to meet our specific system needs results in a system that is both efficient and scalable, and which enables the near-real time collaboration of communications planners.

In the next chapter, we discuss requirements for interoperability between disparate communications planning and monitoring systems, and present a technique that reduces the amount of manual labor involved in the repetition of redundant processes among systems. Reduction of this redundancy through the enabling of information sharing is another key feature in creating a more efficient communications planning workflow.

IV. ENABLING INTEROPERABILITY OF COMMUNICATIONS PLANNING SOFTWARE SYSTEMS

A. BACKGROUND

Software applications currently used by Marine Corps communications planners and operators can be divided into two distinct categories: network modeling and simulation, and network monitoring. Network modeling and simulation software takes input from the user such as geographical terrain data and transmission system specifications and predicts the ability of equipment to transmit effectively between two locations. Network monitoring software assists in the optimization of a network, maintains real-time network status, and assists in troubleshooting network problems.

The applications used to perform the functions described above represent the multiple software tools used by the Marine Corps throughout the process of the development, analysis, implementation, and monitoring of networks. There are four significant problems with these software tools:

- Limited ability to exchange data between them (automatically or manually).
- Lack of standardization across applications, which results in incompatibilities when attempting to exchange information between them.
- PC based rather than server based, limiting access to the information to the local PC/users.
- Information must be manually entered into some systems.

Based on our technical experience and use of the above mentioned tools, we find that the above problems result in significant duplication of effort and an increased vulnerability to human error in the current sequential process flow shown in Figure 16.

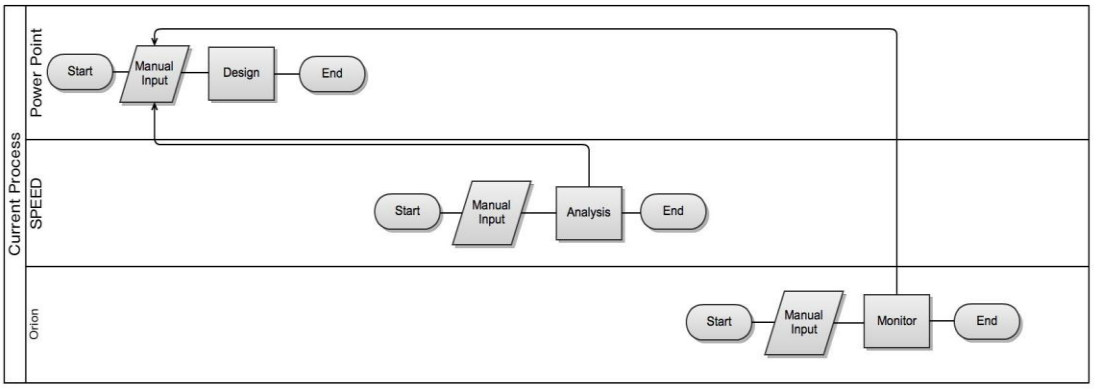


Figure 16. Information flow of current communications planning process

1. Scope

The focus of this chapter is to present a viable approach to addressing interoperability of legacy communications planning systems. We will explore the commonly accepted information-oriented approach. Information-oriented integration typically deals with simple data exchanges between two or more systems; this can be accomplished either through data replication, data federation, or interface processing. The use of enabling technologies such as an integration server/repository, Extensible Markup Language (XML) and Extensible Stylesheet Language Transformations (XSLT) will also be explored. Through literature surveys and some limited experimentation, we investigated the use of the above tools to demonstrate how they could provide a valid and useful means for solving the problems outlined in the background section by enabling interoperability between legacy systems, as shown in Figure 17, which represents the proposed overlapping process flow. We will present a network planning and operations taxonomy as a means to achieve interoperability between the network planning, analysis, and monitoring systems.

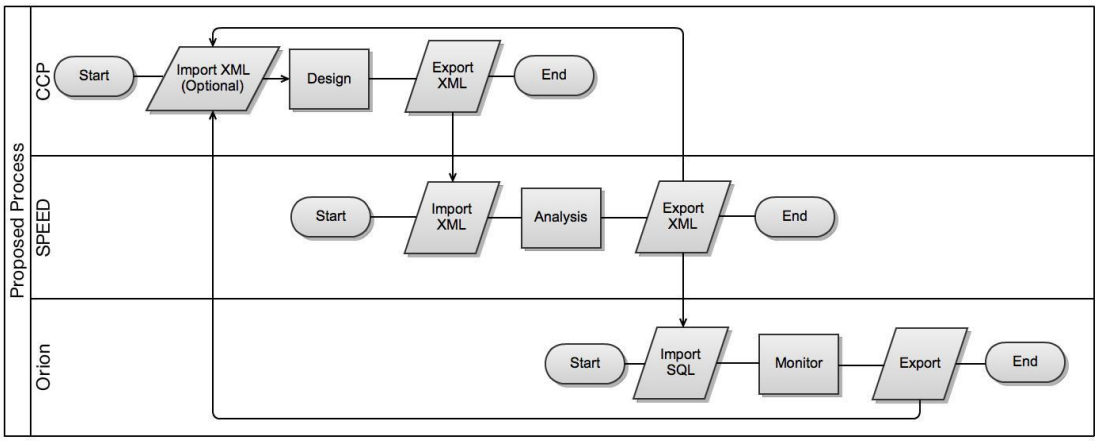


Figure 17. Information flow of proposed communications planning process

B. INTEROPERABILITY

The industry standard meaning of interoperability as defined by IEEE is:

- "the ability of two or more systems or components to exchange information and to use the information that has been exchanged. *See also:* **compatibility**" [21].

Compatibility is defined as:

- "(1) The ability of two or more systems of components to perform their required functions while sharing the same hardware or software environment. (2) The ability of two or more systems or components to exchange information" [21].

The Department of Defense defines interoperability as:

- "the ability of systems, units or forces to provide data, information, materiel, and services, to and accept the same from other systems, units, or forces and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together" [22].

In addition to these definitions, it's common to look at interoperability from three distinct levels: physical, syntactic, and semantic.

- Physical interoperability is the ability to connect systems at the level of electrical signals, such as USB to USB.
- Syntactic interoperability is the ability to communicate with another system through data exchange of a common format. A prerequisite of this is to identify common data elements and communication protocols between systems. XML and SQL are standards among the available tools to accomplish this.
- Semantic interoperability is the ability to share unambiguous meaning. To do this, systems must refer to a common information exchange reference model. An example of this would be the creation of a vector data model using a system like Global Information Network Architecture (GINA) [23].

These definitions provide a baseline understanding of what we're trying to achieve as we explore this project and DoD's doctrine of Net-centric warfare necessitates effective collaborative communications planning and monitoring. Net-centric warfare depends on a seamless exchange of data from various systems. In 2006, the DoD Chief Information Officer published the "Net-Centric Services Strategy" to provide guidance for evolving the DoD net-centric environment to a Service Oriented Architecture. In the document, the DoD states that:

"As the threats facing the DoD evolve, and as new threats begin to emerge, a new level of responsiveness and agility is required from our forces. The DoD cannot transform its operations to support a net-centric force by merely maintaining and expanding the status quo. Patching stovepipes together is a temporary solution; however, this leads to a fragile environment, which will eventually crumble under the high demands and unpredictable needs of the users. The current DoD network consists of information silos that cannot

communicate with each other unless they are pre-wired to do so. In addition, these silos cannot scale to accommodate the levels of interaction that will exist. The DoD's current stovepipe-based information environment must shift to a more robust and agile information environment that can support and enable net-centric operations" [24].

C. OVERVIEW OF ENABLING TECHNOLOGY FOR LEGACY SYSTEMS INTEROPERABILITY

1. Information Oriented Approach

Information-oriented integration deals with data exchanges between multiple systems, accomplished through data replication, data federation, or interface processing. Data replication moves data from system to system with no changes. Data federation integrates data into one master system. Interface processing uses Application Programming Interfaces (API) to integrate multiple applications. Once you determine the type of data exchange required for your domain, you identify the candidate systems and determine what information from them is required for functionality [25].

2. Integration Server/ Repository

An integration server is commonly used in today's industry to facilitate interaction between applications, both internally and externally networked. They provide a number of benefits when it comes to dealing with differences in application semantics, data base schemas, and data transformations. They act as a bridge between different platforms and applications routing information through a number of interface mechanisms. They provide a number of extremely helpful services such as: transformation, intelligent routing, rules processing, message warehousing, flow, control, repository services, directory services, management APIs, and adapters [26]. An example of a transformation service might be that system A uses dates in "9/10/72" format, i.e., mm/dd/yy, while system B uses "September 10, 1972," an integration server would facilitate the conversion from one system to the other, perhaps through an indexed table. The

biggest advantages of integration servers are that they allow the existing systems to stay in place, both physically and with respect to ownership, and they provide the means to exchange data indirectly through the integration server or repository.

3. XML

Standards are an essential in defining common elements for interfaces, representation of data, and protocols for data exchange. They help in achieving interoperability because they are widely accepted by vendors and they increase the likelihood that diverse systems from various sources will be able to exchange useful information. While adherence to standards doesn't guarantee interoperability, it's a step in the right direction. For this project we explored and applied two industry standards: Extensible Markup Language (XML) and Extensible Stylesheet Language Transformations (XSLT).

Programming languages, in our opinion, are created with two main goals in mind: to be unambiguous and to facilitate the creation of reliable, human readable programs. With today's rapidly changing and evolving computing environment, they must also be interoperable to stay relevant. Given the number of programming languages currently in existence, the challenge of achieving interoperability seems unlikely. However, through the use of XML the individual programming languages themselves no longer need to be interoperable with each other, they simply need to use the XML standards developed by the World Wide Web Consortium (W3C) to exchange data. XML was created so that highly structured documents could be used over the web, it's the core building block for a wide range of other technologies, and it enables interoperability.

Alternatives to XML are Hyper Text Markup Language (HTML) and Extensible Hyper Text Markup Language (XHTML); both have their advantages and disadvantages but neither of these standards were suitable for this project because the systems being studied currently use XML. XML, HTML, and XHTML are all based on Standard Generalized Markup Language (SGML), an

International Organization for Standardization (ISO) standard for markup languages developed in 1986. Figure 18 shows the relationship between these standards and the other standards discussed in this project.

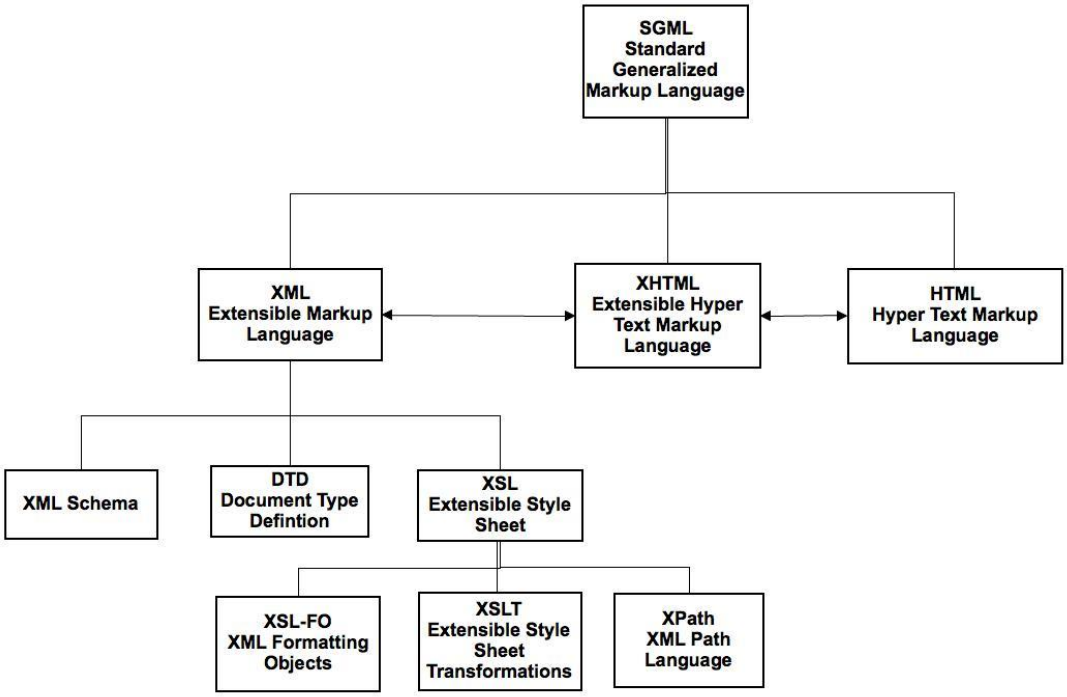


Figure 18. Relationship between SGML, XML, HTML, and XHTML

XML is a markup language, meaning that information about the data is embedded in the document with the data itself. As a result of this, XML is self-describing; you can easily determine what the data is representing, for example a person's name might be expressed clearly as a tag that looks like <name>; this is one of the key attributes that makes it easy to understand and great for data interchange. The main characteristics that set XML apart from the other markup languages are: its emphasis on descriptive rather than procedural markup, its document type concept, and its independence from any one hardware or software system [27]. XML can be used for a wide variety of things, such as document representation (e.g., vector graphics), data structure (e.g., database), and message exchange (e.g. SOAP, WSDL). The basic structure of XML consists of three main components: XML Document Type Definition (DTD), a

grammar for describing the structure of the data; XML Schema, a set of rules that represents how an XML document models its data and defines its elements, attributes, and relationships between elements; XSL (Stylesheet), a family of languages that specifying the formatting and styling of an XML document.

XML is widely recognized as an enabler for system interoperability and integration because it can be used as a data transport format within and between applications or it can be used as a data bridge within and between applications. The Department of Navy (DON) is well aware of this and its guidance since 2002 regarding Extensible Markup Language has been "to fully exploit XML as an enabling technology to achieve interoperability in support of maritime superiority" [28]. Even more currently, in 2009 the USD AT&L issued the following guidance in its report, "to maximize DoD's ability to utilize commercial technology, all commercial vendors should consider the incorporation of Suite B (including the use of extensible markup language (XML) standards) in their products" [29].

4. XSL

Extensible Stylesheet Language is a family of languages that specify the formatting and styling of an XML document. It was initiated to bring the functionality of Document Style Semantics and Specification Language (DSSSL) to XML. In December 1997, the W3C established a working group for XSL and it produced a working draft specification on 18 August 1998. This specification is split into three parts:

- XSLT (Transformation): Designed to be used independently of XSL; it describes how a document is transformed into another document using a set of rules. It is commonly used to translate between different XML schemas, convert XML data into another document, or create printed output through XSL Formatting Objects. XSLT became a W3C "recommendation" on 16 November 1999.
- XSL-FO (Formatting Objects): A markup language for specifying formatting semantics. There are two parts to working with XSL-FO: an

XML document and a means of transforming that document into an output format (readable, printable, or both, e.g. PDF, Braille) [30]. XSL-FO became a W3C recommendation on 15 October 2001.

- XPath (XML Path Language): A query language for selecting specific parts of an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document [31]. XPath became a W3C recommendation on 16 November 1999.

5. XSLT Processor

The XSLT processor takes an XML document and an XSLT stylesheet, and produces a new document using the rules provided by the XSLT stylesheet, as represented in Figure 19. The XSLT stylesheet, along with XPath, provide the instruction set for the processor to produce the resultant document. While the processor can be implemented on either the client-side or server-side it is most commonly done on the server-side through a standalone processor, such as Apache's Xalan or Saxon XSLT, or a component of software.

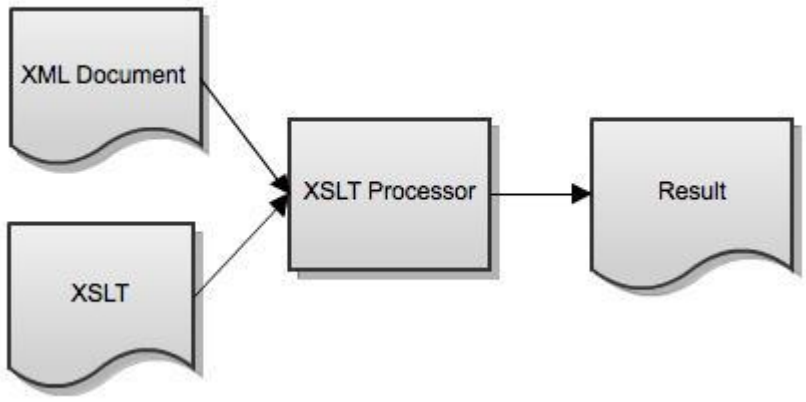


Figure 19. XSLT conversion process

D. SAMPLE APPLICATION OF THE ENABLING TECHNOLOGY

1. Information-Oriented Approach

We believe these enabling technologies are a viable approach to addressing legacy system interoperability and integration of communication planning and analysis tools. This section focuses on the proposed application of these tools as related to our specific problem. We specifically addressed syntactic interoperability and used interoperability as defined by the IEEE to help form the problem space. We determined that our best approach was an information-oriented one that focused on data federation. A few reasons for this were: all the systems being explored import/export XML, they only need to exchange existing data, there didn't appear to be any complex behavior, and this approach was the most direct when compared to other approaches.

The first step in the process of data integration is to identify the data. We begin by identifying the candidate systems and determining what information is required for each system to function properly. A network planning and operations taxonomy as proposed in section E of this chapter provides us the ability to cross reference synonymous terms in the domain and accurately determine what data is required. During this process we can determine what information is common, this includes both commonly named and commonly understood. Additionally, we need to identify any information that may need to be generated by an intermediate source in order to allow the next system to carry out its tasks. Finally, because the systems we are studying are XML based, we can map common data through the schemas associated with them and identify any additional data deficiencies. For this research, we used data models from the Network Planning and Operation Domain, a Cloud-based Collaborative Planning (CCP) tool developed in conjunction with this project, and the Systems Planning Engineering and Evaluation Device (SPEED), a Government off-the-shelf (GOTS) program-of-record. Figure 20 represents an abstract view of the common data elements that can be passed directly from each system, the elements that must be interpreted from one system to the other using an

intermediate source, and any additional elements that may be required by the receiving system. This overall picture serves as incentive for exploring this project because it shows that through data integration and manipulation, we can optimize workflow and use the data more efficiently.

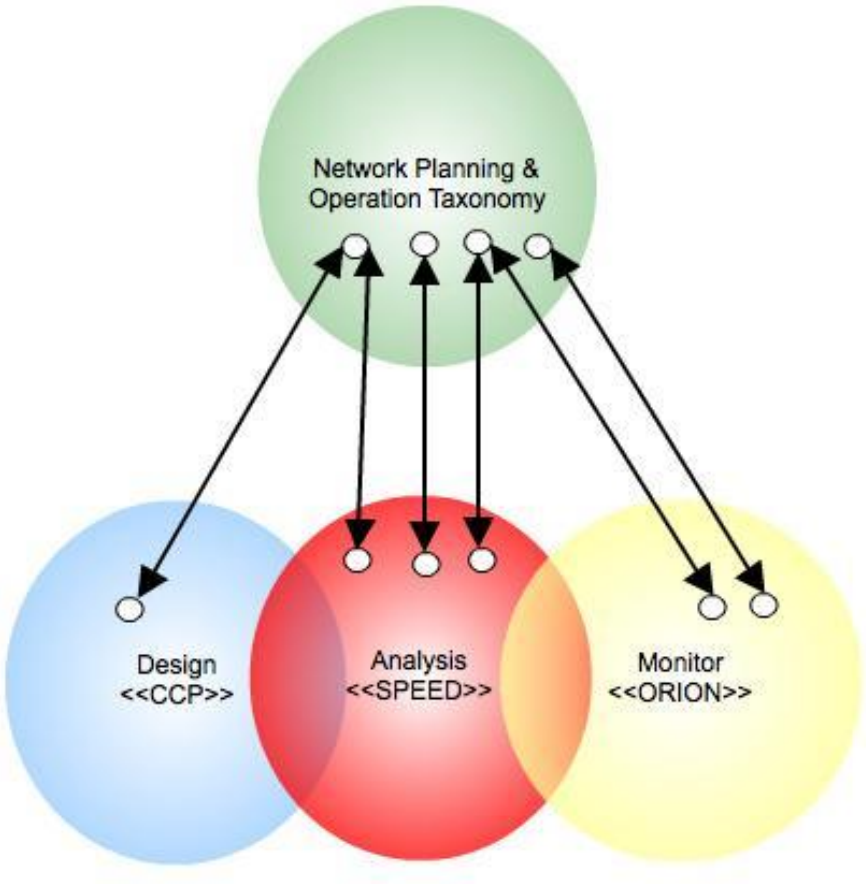


Figure 20. Points of commonality, translation, and addition

The second step in the process of data integration is to determine what information is required for functionality. We did this by taking a look at the steps involved with each process and extracting the data points that were either common, need to be some translated to some degree, or need to be created. The processes are described below.

The basic process flow for designing a network in CCP:

1. Select the menu option to create “New Project.”
2. Select the menu option to create “New Site.”
    a. Set “Site Name”
    b. Set “Unit name.”
    c. Set “Location.”
    d. Position Site on diagram.
3. Select the menu option to create a Link.
    a. Set “Link Designator.”
    b. Set “Start Site” and “End Site.”
    c. Select Link “Equipment.”
    d. Set “Bandwidth.”
    e. Set “Start Location” and “End Location” (Latitude/Longitude).
4. Select the menu option to “Add Circuit.”
    a. Set “Circuit Designator.”
    b. Select “Link” that carries Circuit.
    c. Select “Type” (Transmission, Data, Wire, etc).
    d. Select “Equipment.”
    e. Set Circuit “Bandwidth.”
5. Repeat Steps 2–4 as required.

The basic process flow for analyzing a network in SPEED:

1. Move to proper map area.
2. Place new System.
    a. Right click “Systems,” select “New.”
    b. Select “Radio Type.”

c. Set “Name.”

d. Select position.

3. Set system properties.

    a. Right click “System,” select “Properties.”

    b. Modify “Frequencies,” “Antenna,” “Environment,” etc.

    c. Select “Cryptographic Type,” if applicable.

4. Create Radio Nets.

    a. Right click, “Radio Nets,” select “New.”

    b. Set “Name” and choose “Participating Systems.”

    c. Assign net “Control System,” as required.

    d. Change “Frequency,” as required.

5. Place new Military Unit.

    a. Right click/place “New.”

    b. Set “Name” and select position.

    c. Select unit “Properties.”

    d. Assign “Participating Systems” to unit.

6. Analyze Nets.

    a. Select “Nets/Analyze Nets.”

    b. Select “All Nets” to be analyzed.

    c. Click “Analyze.”

Finally, we used the schemas associated with the systems to map common data and determine if there were any additional data requirements for the receiving system that the sending system may not have provided. Figure 21 represents a snippet of the CCP schema file, the actual file can found in Appendix H (CCP SCHEMA).

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="project">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="observers" type="xsd:string" />
        <xsd:element name="name" type="xsd:string" />
        <xsd:element name="creationDate" type="xsd:string" />
        <xsd:element name="modifiedDate" type="xsd:string" />
        <xsd:element name="diagramCanvas" type="xsd:string" />
        <xsd:element name="sites">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="site">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="ID" type="xsd:string" />
                    <xsd:element name="name" type="xsd:string" />
                    <xsd:element name="unitName" type="xsd:string" />
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
```

Figure 21. Snippet of the CCP schema file

2. Integration Server/ Repository

For the purpose of this research, we explore the use of a repository and defer the use of an actual integration server as future work. A repository is central storage location that contains information about source and target applications. It provides a means to maintain information about message schema, metadata, enabling technology, transformation, processing rules/logic, etc., and it will serve as a master directory for the communications planning domain, linking all the systems to each other.

The repository will need to be able to join equivalent data and maintain application specific rules. An example of why this is necessary can be seen in the following: SPEED uses a hash as a key value (e.g. 0DF0CEF1–6E60–4566-A8E2–0E9A805F2C79) when referring to equipment and CCP uses a standard naming convention such as MRC-142C. A repository would maintain a master table that would link these two data elements. A design possibility is to maintain this table by a key value of National Stock Number (NSN); this will allow a smooth integration with other systems outside of the communications domain without requiring a fully-meshed naming correlation. Table 1, represents the repository and shows how the information is tied together.

National Stock Number	Nomenclature	Equipment Name (CCP)	System SystemTypeID (SPEED)	SystemType (SPEED)
5820–01–545–6691	Digital Wideband Transmission System AN/MRC-142C	MRC-142C	0DF0CEF1–6E60–4566-A8E2–0E9A805F2C79	MRC-142C
5895–01–354–7601	Radio Terminal Set, AN/TRC-170 (V5)	TRC-170 (V5)	B9C77F7B-06D4–48A5–9DF6–728FEC99772A	TRC-170 (V5)

Table 1. Mapping CCP equipment to SPEED equipment in repository

The repository is responsible for maintaining schema information and transformation style sheets; however, an external processor (e.g. Saxon) will do the actual transformation, as outlined later.

3. XML

The use of standards was essential to the success of this project. All the systems being explored imported and exported XML; this simplified the integration process. The following discussion will define the XML elements along with their attributes and provide examples of the XML files used in CCP and SPEED, as defined in the SPEED XML Description Document provided by Northrop Grumman. Figure 22 is the basic structure of a CCP XML File and Table 2 is the definition of each of the elements used in the file.

```
<project>
  <observers/>
  <name>test_project</name>
  <creationDate>2012-03-22 07:54:47.267 UTC</creationDate>
  <modifiedDate>2012-03-22 07:58:44.428 UTC</modifiedDate>
  <diagramCanvas>C4769D2D-B79E-4480-B903-B63512B7E553:160,60,150,195/E6AFF14C-5B51-4F12-802
  <sites>
    <site>
      <ID>C4769D2D-B79E-4480-B903-B63512B7E553</ID>
      <name>Red Beach</name>
      <unitName>CLR-17</unitName>
    </site>
    <site>
      <ID>2ADFA69D-2AFC-448C-9001-32E8E109ECCF</ID>
      <name>Las Pulgas</name>
      <unitName>2/11</unitName>
    </site>
  </sites>
  <links>
    <link>
      <ID>E6AFF14C-5B51-4F12-8025-0BEC9C7567DE</ID>
      <linkDesignator>MZZ-001</linkDesignator>
      <startSite reference="../../../sites/site"/>
      <endSite reference="../../../sites/site[2]"/>
      <startLocation>
        <latitude>32.2212027777778</latitude>
        <longitude>-116.914866666667</longitude>
      </startLocation>
      <endLocation>
        <latitude>32.0923</latitude>
        <longitude>-116.885127777778</longitude>
      </endLocation>
      <equipment>
        <ID>0DF0CEF1-6E60-4566-A8E2-0E9A805F2C79</ID>
        <name>AN/MRC-142C</name>
        <imageSrc>mrc142c.png</imageSrc>
        <type>Trans</type>
      </equipment>
      <bandWidth>512</bandWidth>
      <circuits/>
    </link>
  </links>
</project>
```

Figure 22. CCP XML File

	Description
Project	The root element for the entire Project.
Observers	Auto-generated element identifying the use of the Observer Pattern to identify changes to the project.
Name	The user assigned name of the project.
CreationDate	The date the project was created.
ModifiedDate	The date the project was last modified.
DiagramCanvas	An auto-generated unique ID.
Sites	The root element for all Site instances contained in CCP.
Site	A particular area of interest.
ID	A unique identifier assigned to a site.
Name	The name assigned to this Site by the user
UnitName	The name assigned to this Unit by the user.
Links	The root element for all Link instances contained in CCP.
Link	A piece of equipment used to link two nodes.
ID	A unique identifier assigned to a Link.
LinkDesignator	The Link Designator as defined by MCWP 6–22.
StartSite	Identifies the starting site in this network.
EndSite	Identifies the ending site in this network.
StartLocation	Identifies the starting location of the Equipment.
Latitude	A geographic coordinate that specifies the north-south position of a point on the Earth's surface.
Longitude	A geographic coordinate that specifies the east-west position of a point on the Earth's surface.
EndLocation	Identifies the ending location of the Equipment.
Equipment	Devices facilitating the use of a computer network (e.g., gateways, routers, network bridges, switches, hubs, repeaters, multilayer switches, protocol converters, bridge routers, proxy servers, firewalls, network address translators, multiplexers, network interface controllers, wireless network interface controllers, modems, ISDN terminal adapters, line drivers, wireless access points, networking cables, etc.).
ID	A unique identifier assigned to Equipment.
Name	The name of the Equipment as defined by the FEDLOG System.
ImageSrc	The image source files used by CCP to graphically display the equipment.
Type	Identifies the specific type of equipment (e.g., transmission).
Bandwidth	A rate of data transfer, bit rate or throughput, measured in bits per second (bps).
Circuits	A dedicated communications channel for the links to communicate that guarantees bandwidth.

Table 2. CCP CML file definitions

Figure 23 is an example of a basic SPEED XML File. The SpeedXMLRun is the root node, followed by the SPEED Path Profiler Model (SPPModel) node. Both of these nodes are placeholders and merely enforce the document structure. The Document node has four child nodes; each of the child nodes will be further defined in the below figures and tables. For the purpose of this project we will not need to use or discuss the MapCenter, Graphics, ActiveDataSets, MapModel, SnapIns and ObjectGraphics elements.

Figure 23. SPEED XML File

The first of the of Document's child nodes is the Systems node. A system can be a defined as a platform or radio, Table 3 defines the System Attributes. When a system is created, a new Global Unique Identifier (GUID) is generated and saved. This GUID ensures that the new system is uniquely identified. The SystemTypeID refers to the system type that can be found in the SPEED Database; this database contains the system type common names (e.g. MRC-142C) and any specifications for the equipment.

	Description
Systems	The Document or root element for all system instances contained in the SPEED run.
System	The main XML element containing the parameters of a system.
ID	The globally unique identifier (GUID) for the actual instance of a system.
Name	The user assigned name for the system instance.
ParentSystemID	The globally unique identifier (GUID) for the actual instance of a system that contains this system instance.
SystemTypeID	The globally unique identifier (GUID) assigned to the type of system.
Notes	The user assigned notes concerning a system instance.
IsActive	For external record keeping of system, set to 1.
Radius	A user assigned value for use in visually displaying a circle on map at a set radius from the system.
IsRadiusVisible	Used to show visibility of radius on or off, set to 1 for on and 0 for off.
MilUnitID	The globally unique identifier (GUID) for the Military Unit associated with the system.
NetID	The globally unique identifier (GUID) for the Net associated with the system.
MapPlaceableID	The globally unique identifier (GUID) for the map graphic object of the system.
Refractivity	The refractivity of the system, valid integer values are 200–450 (inclusive).
GroundType	The ground type of the environment of the system.
Humidity	The humidity type of the environment of the system.
ElectroMag	The electromagnetic noise type for the environment of the system.

Table 3. System attributes (platform) definitions

Figure 24 is an example of a Systems node (Platform) within SPEED Path Profiler Model (SPPModel) Document:

```
<Systems>
    <System ProgID=\"SPP.Platform.1\" clsid=\"{D03BBC73-2106-4989-BC6D-92881FD934E9}\">
        <CustomPropCount vt=\"5\">0</CustomPropCount>
        <ID vt=\"8\">{00A2867D-2B56-411F-9E02-CD5A3164F714}</ID>
        <Name vt=\"8\">Platform0001</Name>
        <ParentSystemID vt=\"8\">
        </ParentSystemID>
        <SystemTypeID vt=\"8\">{0DF0CEF1-6E60-4566-A8E2-0E9A805F2C79}</SystemTypeID>
        <Notes vt=\"8\">
        </Notes>
        <IsActive vt=\"3\">1</IsActive>
        <Radius vt=\"5\">0</Radius>
        <IsRadiusVisible vt=\"3\">0</IsRadiusVisible>
        <MilUnitID vt=\"8\">
        </MilUnitID>
        <NetID vt=\"8\">{5922AC36-8E33-4963-BF55-896CF43CABA4}</NetID0>
        <MapPlaceableID vt=\"8\">{17DF4351-E5D9-46FD-9303-4054549F94C1}</MapPlaceableID>
        <Refractivity vt=\"5\">320</Refractivity>
        <GroundType vt=\"3\">1</GroundType>
        <Humidity vt=\"3\">2</Humidity>
        <ElectroMag vt=\"3\">0</ElectroMag>
    </System>
</Systems>
```

Figure 24. Systems attributes (platform) XML

Table 4 defines the basic radio attributes, denoted “System Attributes (Radio).” This system is separate from platforms because radios contain unique attributes that other systems do not have. Within the SPEED database, radios have several tables associated with each other to complete the system (e.g., type, antenna). The tables use IDs to link the tables.

	Description
TransceiverTypeID	The globally unique identifier (GUID) assigned to the type of Transceiver.
Power	The power for the transmitter of Transceiver.
PowerUnits	The unit of measure for the power of Transceiver.
TXFreq	The frequency of the transmitter for Transceiver.
RXFreq	The frequency of the receiver for Transceiver.
Mode	The selected mode for the Transceiver.
Band	The selected band for the Transceiver.
AntennaHeight	The height of the antenna for the Transceiver.
AntennaTypeID	The globally unique identifier (GUID) for the type of antenna for the Transceiver.
AntennaPolarization	The polarization of the antenna for the Transceiver.
NetID	The globally unique identifier (GUID) for the Net associated with the system.
CryptoType	The type of crypto associated with the Transceiver.
DataRate	The data rate for the Transceiver.
TRCSigQuality	The signal quality for the Transceiver.
Modulation	The current modulation setting.
IsData	Using data setting, set 1 for true, 0 for false.

Table 4. System attributes (radio) definitions

Figure 25 is an example of a Systems node (Radio) within SPEED Path Profiler Model (SPPModel) Document:

```
<Systems>
    <System ProgID=\"SPP.Radio.1\" clsid=\"{D03BBC73-2106-4989-BC6D-92881FD934E9}\">
        <CustomPropCount vt=\"5\">0</CustomPropCount>
        <ID vt=\"8\">{00A2867D-2B56-411F-9E02-CD5A3164F714}</ID>
        <Name vt=\"8\">Radio0001</Name>
        <ParentSystemID vt=\"8\"></ParentSystemID>
        <SystemTypeID vt=\"8\">{0DF0CEF1-6E60-4566-A8E2-0E9A805F2C79}</SystemTypeID>
        <Notes vt=\"8\">
        </Notes>
        <IsActive vt=\"3\">1</IsActive>
        <Radius vt=\"5\">0</Radius>
        <IsRadiusVisible vt=\"3\">0</IsRadiusVisible>
        <MilUnitID vt=\"8\"></MilUnitID>
        <NetID vt=\"8\">{5922AC36-8E33-4963-BF55-896CF43CABA4}</NetID0>
        <MapPlaceableID vt=\"8\">{17DF4351-E5D9-46FD-9303-4054549F94C1}</MapPlaceableID>
        <Refractivity vt=\"5\">320</Refractivity>
        <GroundType vt=\"3\">1</GroundType>
        <Humidity vt=\"3\">2</Humidity>
        <ElectroMag vt=\"3\">0</ElectroMag>
        <TransceiverLinksCount vt=\"3\">0</TransceiverLinksCount>
        <TransceiverTypeID0 vt=\"8\">{DB94726D-A566-46A1-B525-88A7D7A6B31A}">
        </TransceiverTypeID0>
        <Power0 vt=\"5\">2.5</Power0>
        <PowerUnits0 vt=\"8\">W</PowerUnits0>
        <TXFreq0 vt=\"5\">225</TXFreq0>
        <RXFreq0 vt=\"5\">225</RXFreq0>
        <Mode0 vt=\"3\">1</Mode0>
        <Band0 vt=\"3\">1</Band0>
        <AntennaHeight0 vt=\"5\">15.24</AntennaHeight0>
        <AntennaTypeID0 vt=\"8\">{287AB8BC-42CE-4BE9-A784-6D542DFF4B93}</AntennaTypeID0>
        <AntennaPolarization0 vt=\"8\">H</AntennaPolarization0>
        <NetID0 vt=\"8\">{5922AC36-8E33-4963-BF55-896CF43CABA4}</NetID0>
        <CryptoType0 vt=\"8\">
        </CryptoType0>
        <DataRate0 vt=\"3\">256000</DataRate0>
        <TRCSigQuality0 vt=\"8\">1</TRCSigQuality0>
        <Modulation0 vt=\"8\"></Modulation0>
        <IsData1 vt=\"3\">0</IsData1>
    </System>
</Systems>
```

Figure 25. Systems Attributes (Radio) XML

A net is defined as a grouping of radios communicating. Table 5 defines the Net attributes:

Table 5. Net attributes definitions

	Description
ID	The globally unique identifier (GUID) assigned to this Net instance.
Name	The name assigned to this Net by the user or auto-generated by SPEED.
EmmisionDesignator	The emissions designator for the Net.
Band	The band of the net (HF, VHF, UHF, SHF, or EHF).
NetTypeID	The globally unique identifier (GUID) for the

	Net type.
Frequency	The frequency used by the Net.
SOP	SOP for the Net.
Provision	The provision value for the Net.
Restoratioin	The restoration value for the Net.
UpLink	The uplink value for the Net in MHz.
DownLink	The downlink value for the Net in MHz.
NetModeFlag	Flag value for the Net model.
IsTacSatNet	Tactical Satellite Net status set to 1 for True, 0 for false.
TacSatAccessType	Tactical Satellite access type (if applicable).
TacSatBandwidth	Tactical Satellite bandwidth (if applicable).
TacSatDataRateBps	Tactical Satellite data rate (if applicable).
TacSatConfigCode	Tactical Satellite configuration code (if applicable).
TacSatDMarkNum	Tactical Satellite demarcation number (if applicable).
TacSatOW	Tactical Satellite preset channel for OW key (if applicable).
TacSatHomeChannel	Tactical Satellite home channel (if applicable).
TacSatChannel	Tactical Satellite channel (if applicable).
TacSatelliteType	Tactical Satellite type (if applicable).
TacSatGuardAddress	Tactical Satellite guard address (if applicable).
MembersCount	Number of member systems in the Net.
MemberDn	The globally unique identifier (GUID) of a member system. The n represents a sequentially incremented number.
MemberDAndAffiliation	The globally unique identifier (GUID) of a member system flowed by a colon':' and its affiliation 'n' is a sequentially incremented number.
AffiliationCount	Number of distinct affiliation records.
TacSatPropsCount	Number of valid Tactical Satellite properties.
TransceiverMapRadion	The globally unique identifier (GUID) of a member system. The n represents a sequentially incremented number.
TransceieverMapTransceiverIDn	The globally unique identifier (GUID) of a member system's transceiver. The n represents a sequentially incremented number.
TransceiverMapCount	Number of transceivers utilized by the Net.

Table 5. Net attributes definitions

Figure 26 is an example of the Nets node within SPEED Path Profiler Model (SPPModel) Document:

```
<Nets>
    <Net clsid=\"{E9FC9B54-6941-4681-94E1-7B663293536E}\">
        <ID vt=\"8\">{5922AC36-8E33-4963-BF55-896CF43CABA4}</ID>
        <Name vt=\"8\">MZZ03</Name>
        <EmissionDesignator vt=\"8\">106KW1D</EmissionDesignator>
        <Band vt=\"8\"></Band>
        <NetTypeID vt=\"8\"></NetTypeID>
        <Frequency vt=\"5\">225</Frequency>
        <SOP vt=\"8\"></SOP>
        <Provision vt=\"8\"></Provision>
        <Restoration vt=\"8\"></Restoration>
        <UpLink vt=\"8\"></UpLink>
        <DownLink vt=\"8\"></DownLink>
        <NetModelFlag vt=\"3\">2</NetModelFlag>
        <IsTacSatNet vt=\"8\"></IsTacSatNet>
        <TacSatAccessType vt=\"8\">1</TacSatAccessType>
        <TacSatBandwidth vt=\"8\">0</TacSatBandwidth>
        <TacSatDataRateBps vt=\"8\">0</TacSatDataRateBps>
        <TacSatConfigCode vt=\"3\">60</TacSatConfigCode>
        <TacSatDMarkNum vt=\"8\"></TacSatDMarkNum>
        <TacSatOW vt=\"8\"></TacSatOW>
        <TacSatHomeChannel vt=\"8\"></TacSatHomeChannel>
        <TacSatChannel vt=\"8\"></TacSatChannel>
        <TacSatSatelliteType vt=\"8\"></TacSatSatelliteType>
        <TacSatGuardAddress vt=\"8\"></TacSatGuardAddress>
        <MembersCount vt=\"3\">2</MembersCount>
        <MemberID1 vt=\"8\">{00A2867D-2B56-411F-9E02-CD5A3164F714}</MemberID1>
        <MemberID2 vt=\"8\">{CFE04555-5251-4946-8D28-42C75C44EDE3}</MemberID2>
        <MemberIDAndAffiliation1 vt=\"8\">{00A2867D-2B56-411F-9E02-CD5A3164F714}:4
        </MemberIDAndAffiliation1>
        <MemberIDAndAffiliation2 vt=\"8\">{CFE04555-5251-4946-8D28-42C75C44EDE3}:0
        </MemberIDAndAffiliation2>
        <AffiliationCount vt=\"3\">2</AffiliationCount>
        <TacSatPropsCount vt=\"3\">0</TacSatPropsCount>
        <TransceiverMapRadioID1 vt=\"8\">{00A2867D-2B56-411F-9E02-CD5A3164F714}</TransceiverMapRadioID1>
        <TransceiverMapTransceiverID1 vt=\"8\">{DB94726D-A566-46A1-B525-88A7D7A6B31A}</TransceiverMapTransceiverID1>
        <TransceiverMapRadioID2 vt=\"8\">{CFE04555-5251-4946-8D28-42C75C44EDE3}</TransceiverMapRadioID2>
        <TransceiverMapTransceiverID2 vt=\"8\">{A8EC2E71-3802-4F9B-AAE8-0E577BEBF661}</TransceiverMapTransceiverID2>
        <TransceiverMapCount vt=\"3\">2</TransceiverMapCount>
    </Net>
</Nets>
```

Figure 26. Net attributes XML

SPEED supports the placement of military unit symbols on the displayed network. Radios that support the military units can be associated with their respective military unit symbols. Movement of a military unit symbol simultaneously causes the associated radios to move along with the military unit symbol. Table 6 defines the MilUnits attributes:

	Description
ID	The globally unique identifier (GUID) assigned to this MilUnit instance.
Name	The name assigned to this MilUnit by the user or auto-generated by SPEED.
Notes	The user assigned notes concerning a MilUnit instance.
SymbolCode	The character representation for the symbol used for MilUnit icon.
IsCollapsed	Should MilUnit be displayed collapsed, 1 for collapsed, 0 for expanded.
MembersCount	Total number of contained systems.
SymbolSet	The symbol set containing the MilUnit icon graphic.
MapPlaceableID	The globally unique identifier (GUID) for the map graphic object of the system.
MemberIDn	The globally unique identifier (GUID) for the contained system. The n represents a sequentially updated number.

Table 6. MilUnits definitions

Figure 27 is an example of the MilUnits node within SPEED Path Profiler Model (SPPModel) Document:

```
<MilUnits>
    <MilUnit clsid=\"{16B82300-7241-4034-A343-13DEC21B110C}\">
        <ID vt=\"8\">{17DF4351-E5D9-46FD-9303-4054549F94C1}</MapPlaceableID>
        <Name vt=\"8\">A Co</Name>
        <Notes vt=\"8\">
        <SymbolCode vt="8">MARFOR</SymbolCode>
        <IsCollapsed vt="3">1</IsCollapsed>
        <SymbolSet vt="3">1</SymbolSet>
        <MembersCount vt=\"3\">0</MembersCount>
        <MapPlaceableID vt=\"8\">{17DF4351-E5D9-46FD-9303-4054549F94C1}</MapPlaceableID>
    </MilUnit>
</MilUnits>
```

Figure 27. MilUnits XML

4. XSLT

The first step in defining a style sheet is to map the existing common data elements. For example, the <site> <name> in CCP needs to map to <System><Name> in SPEED. Knowing this enables us to map the movement from source system to target system. In addition to the data mapping, we need to

track where the information is physically located, any security restrictions, what enabling technology is used, and what data transportation requirements are required. In this particular case, we are keeping the problem space relatively small by using only two systems, so information can be kept locally on the same computer, thereby mitigating security restrictions; XML is the enabling technology for both systems, and data transportation isn't required.

Table 7 is a complete list of common data mappings:

CCP	SPEED
<site> <name>	<System><Name>
<equipment><ID>	<System><ID>
<site><unitName>	<MilUnit><Name>
<link><ID>	<Net><ID>
<link><linkDesignator>	<Net><Name>
<link><location><latitude>	<MapPlaceable><Latitude>
<link><location><longitude>	<MapPlaceable><Longitude>
<link><bandWidth>	<System><DataRate>

Table 7. CCP to SPEED data mappings

Table 8 is a list of additional data requirements for processing by XSLT, these elements exist in both systems but are represented by different names. They simply need to be identified and mapped to the correct target element.

CCP	SPEED
<equipment> <name>	<System><SystemTypeID>

Table 8. Additional SPEED data requirements handled by XSLT

Table 9 is a list of additional SPEED data requirements for processing by XSLT, these elements exist only in SPEED and are required for the XML file to process. For this project, we addressed this by creating the unique identifier in CCP and passing the value through the CCP XML file. In the future, this functionality would be better represented as either a plugin that resides on the repository for each system that requires a hash code or implemented through the XSLT hashing function.

CCP	SPEED	NOTES
N/A	<MilUnit<ID>	Randomly Generated Hash
N/A	<MapPlaceable><MapPlaceableID>	Randomly Generated Hash

Table 9. Additional SPEED data requirements for SPEED processing

The result of the above data mapping aids us in producing an XSLT Style Sheet. Figure 28 represents a snippet of the CCP to SPEED Style Sheet file.

```
<?xml version='1.0' encoding='utf-8'?>
    <xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
        <xsl:output method='xml' version='1.0' encoding='utf-8' indent='yes'/>
        <xsl:template match="/">
            <SpeedXMLRun>
                <UDSystemTypes>
                    <UDSystems/>
                </UDSystemTypes>
                <SPPModel>
                    <SPEEDVer/>
                    <MinClassificationBanner/>
                    <DocGUID/>
                    <Document>
                        <ClassificationBanner/>
                        <DefaultFadeMargin/>
                        <Systems>
                            <xsl:for-each select="project/sites/site">
                                <System>
                                    <Name>
                                        <xsl:value-of select="unitName"/>
                                    </Name>
                                </System>
                            </xsl:for-each>
                        </Systems>
                    </Document>
                </SPPModel>
            </SpeedXMLRun>
        </xsl:template>
    </xsl:stylesheet>
```

Figure 28. Snippet of the CCP to SPEED Style Sheet

An abstract view showing how all the enabling technologies work together is provided in Figure 29.

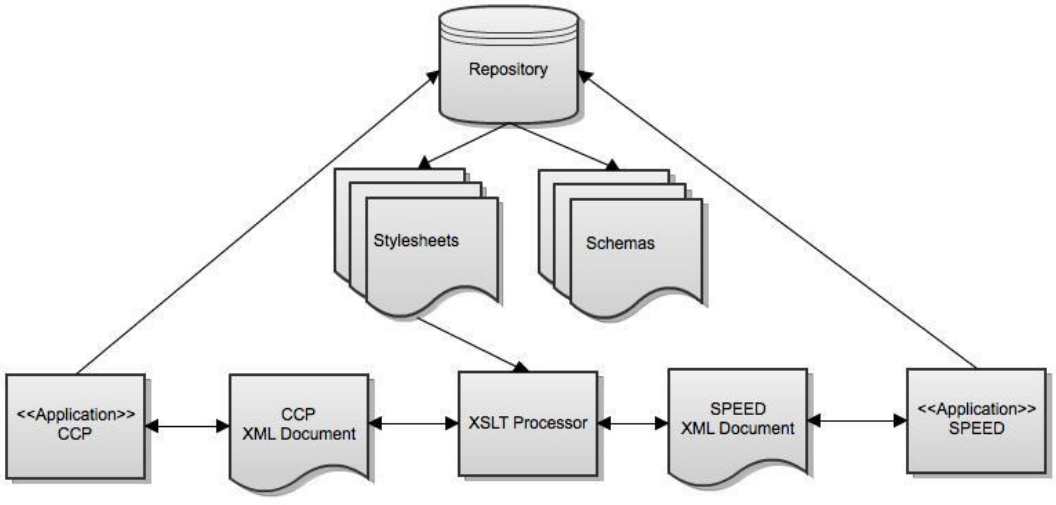


Figure 29. Enabling Technologies Applied

E. NETWORK PLANNING AND OPERATIONS TAXONOMY

Taxonomy is the practice and science of classification; it can be used as a means to achieve interoperability by identifying commonality between different domains and defining the relationships between different entities. Ultimately, it provides a clear understanding of the domain and shows a common representation of all the smaller domains within it.

In this section we present a taxonomy for network planning and operations, as shown in Figure 30. The base for this taxonomy are the four essential characteristics for a MAGTF Communications Network, as defined by Marine Corps Warfighting Publication 3–40.3. These characteristics are transmission, service, multiplexing, and switching, shown underlined in the diagram. A tactical communications network is the technical means by which different methods of communication are linked together. The network is designed to satisfy information exchange requirements and provide access to networked services such as voice, imagery, and data. A network consists of at least one or multiple transmissions between various sites and equipment. A transmission has several characteristics that will be defined during the Marine Corps Planning Process (MCPP). One of these characteristics is service, the method by which a user interacts with information and how information is presented, accessed, used, and exchanged. A service can either be provided through a single signal or through combining two or more discrete signals into a single, higher capacity signal (multiplexing). Another characteristic is the type of transmission media used, either guided (wired, cable, or fiber) or unguided (wireless). This characteristic will be further defined by equipment. The **bold** entities in Figure 30 (e.g. SHF) represent the use of equipment; those pieces of equipment will later be represented by the actual data in the XML files. The key to this taxonomy is in its ability to show commonality between the terminologies of different systems. We show this in Figure 30, by providing the *aliases* that may be used by the other systems involved (e.g., a service will be referred to as a circuit in the design

process). This maintains the integrity and understanding of the basic communications network structure from domain to domain.

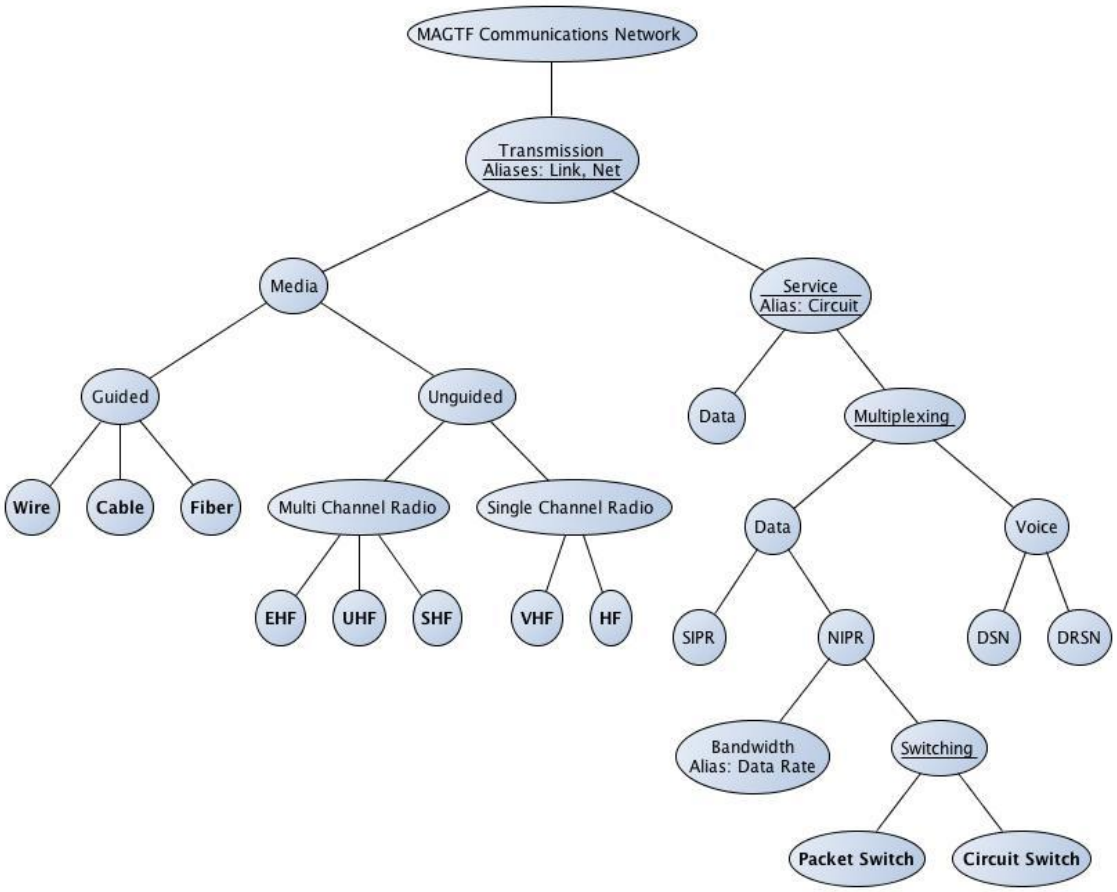


Figure 30. Network Planning and Operations Taxonomy

Table 10 provides the definitions of the terminology used in Figure 30.

Table 10. Network planning and operations taxonomy definitions

	Description
MAGTF Communications Network	The Marine Air Ground Task Force (MAGTF) communications network supports information exchange requirements—voice, data, video, and imagery—both internal and external to the MAGTF.
Transmission	Transmission is the process of conveying a signal from point to point along a path.
Guided	Guided transmission media use physical conductors.
Wire	The most widely used medium for telecommunication (e.g.,

	twisted pair wire).
Cable	Widely used for cable television systems, office buildings, and other work-sites for local area networks (e.g., coaxial cable).
Fiber	A glass fiber that uses pulses of light to transmit data (e.g., fiber optic).
Unguided	Unguided transmission media use electromagnetic waves that propagate through the atmosphere, but are not guided down a specific path.
Single Channel Radio	Single Channel Radios typically operate at half-duplex, meaning that a user may transmit or receive at any given instant, but may not do both simultaneously. They primarily provide the ability to exchange voice, but may also exchange data.
HF	High Frequency 2–9.9999 MHz
VHF	Very High Frequency 30–88 MHz
Multi-Channel Radio	Multi-Channel Radios operate at full-duplex; information can be transmitted and received simultaneously. They provide multiple channels over a single pathway, accommodating multiple users and services simultaneously.
UHF	Ultra High Frequencies 225MHz-3GHz
SHF	Super High Frequencies 3GHz – 30GHz
EHF	Extremely High Frequencies 30–300GHz
Multiplexing	Multiplexing is the process of combining two or more discrete signals into a single, higher capacity signal.
Data	Data is information.
Bandwidth	The difference between the limiting frequencies of a continuous frequency band expressed in hertz (cycles per second). The term bandwidth is also loosely used to refer to the rate at which data can be transmitted over a given communications circuit. In the latter usage, band- width is usually expressed in either kilobits per second or megabits per second.
NIPR	Non-Secure Internet Protocol Router Network used for unclassified information.
SIPR	Secure Internet Protocol Router Network used for classified information (up to SECRET).
Switching	Switching provides the ability to connect many users and their terminal devices in a way that permits on-demand exchange with other users and terminal devices without having to link them individually.
Circuit Switch	Circuit switching is the process of interconnecting a specific circuit to provide a direct connection between calling and called stations, and is historically used for telephone

	networks.
Packet Switch	Packet switching shares a communications path, information is broken up into smaller units, or packets, that are routed to the destination independent of each other. At the receiving end, the packets are reassembled into the original information.
Voice	Communication by word of mouth.
DSN	Defense Switched Network used for unclassified voice network.
DRSN	Defense Red Switched Network used for classified voice network.

Table 10. Network planning and operations taxonomy definitions

An example using this taxonomy along with the transmission diagram provided in Appendix A is shown in Figure 31. The transmission diagram has three Microwave links but for this example we will focus on System Link Designator (SLD) MZZ03, which has two sites and one transmission element. Alpha Company, at Red Beach 1, needs to establish a communication link consisting of various services (alias: circuits) with Alpha Forward, at Red Beach 2. The services required are a NIPR data circuit, which will operate at a bandwidth (alias: data rate) of 512 Kbps, a SIPR data circuit which will operate at a bandwidth (alias: data rate) of 512 Kbps, a Digital Trunk (or Transmission) Group (DTG) voice circuit which will operate at a bandwidth (alias: data rate) of 288 Kbps, and a partial-T1 voice circuit which will operate at a bandwidth (alias: data rate) of 144 Kbps.

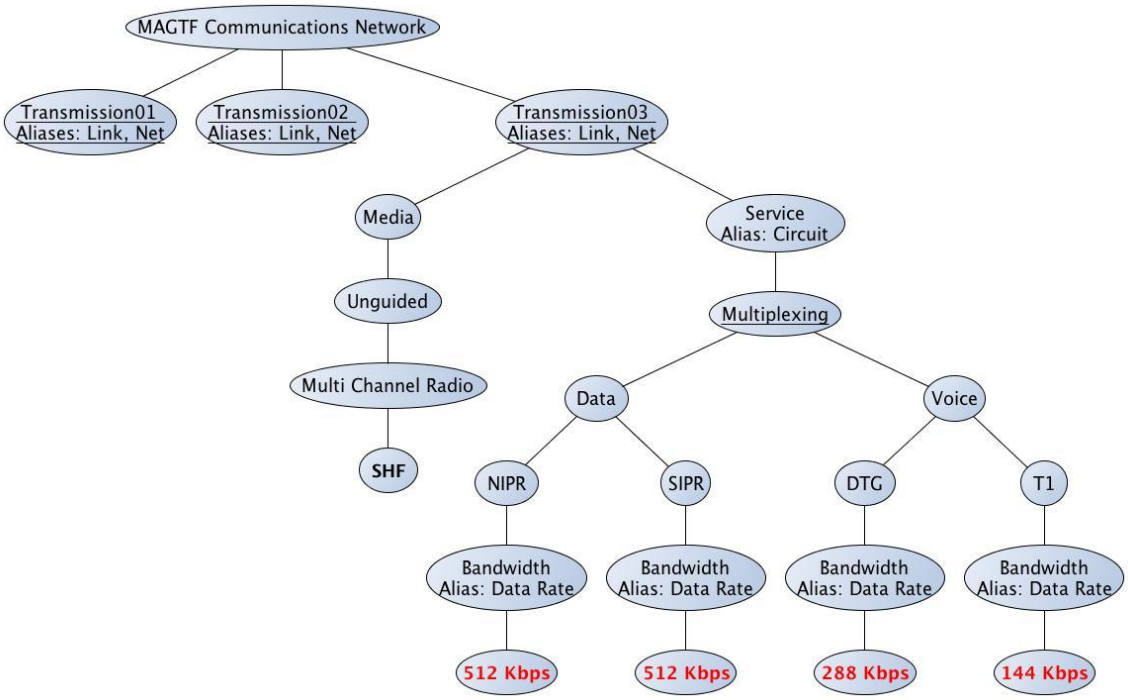


Figure 31. Taxonomy Example Showing Three Microwave Links (Service Alias: Circuit)

This communication link is physically implemented using equipment (alias: SystemType). A piece of equipment has several characteristics, both unique and common characteristics. Unique characteristics are those that are particular to a unique and specific piece of equipment (e.g. Serial Number: 1234), as shown in Figure 32, and common characteristics are those that are common to devices of a particular family, such as all TRC-170s (e.g. Spectrum: Microwave Frequency between 4.4 and 50 GHz), as shown in Figure 33. This example uses a TRC-170 at both locations, which is an unguided, Super High Frequency, Multi-Channel Radio that uses a digital signal to transmit information. Additionally, it provides a multiplexing capability allowing both data and voice communications to use the same equipment, assuming appropriate traffic load management. As stated earlier, the key to this taxonomy is in its ability to show commonality between the terminologies of different systems, the above example will assist us in understanding how mapped items relate to one another from one system to

the next, ensuring that the meaning of the data remains the same as it is exchanged. Using Figure 32 as an example, we clearly see that a TRC-170 in CCP is referred to as Equipment while in SPEED it is referred to as SystemType, knowing and physically seeing this in our taxonomy validates that our process of mapping items is correct as seen in Table 1.

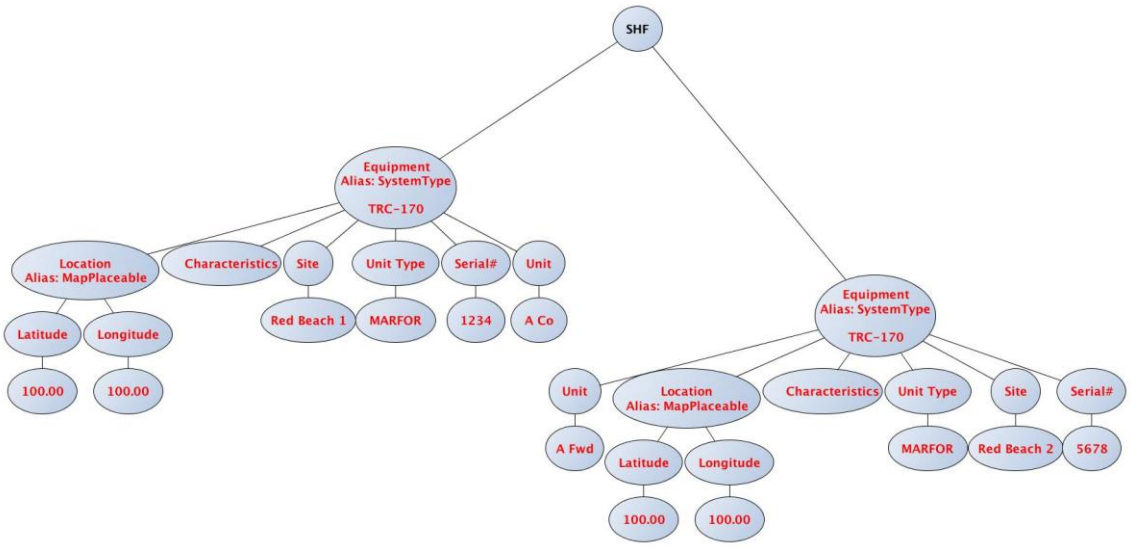


Figure 32. Network Planning and Operations Taxonomy (Equipment Alias: SystemType) Unique Characteristics

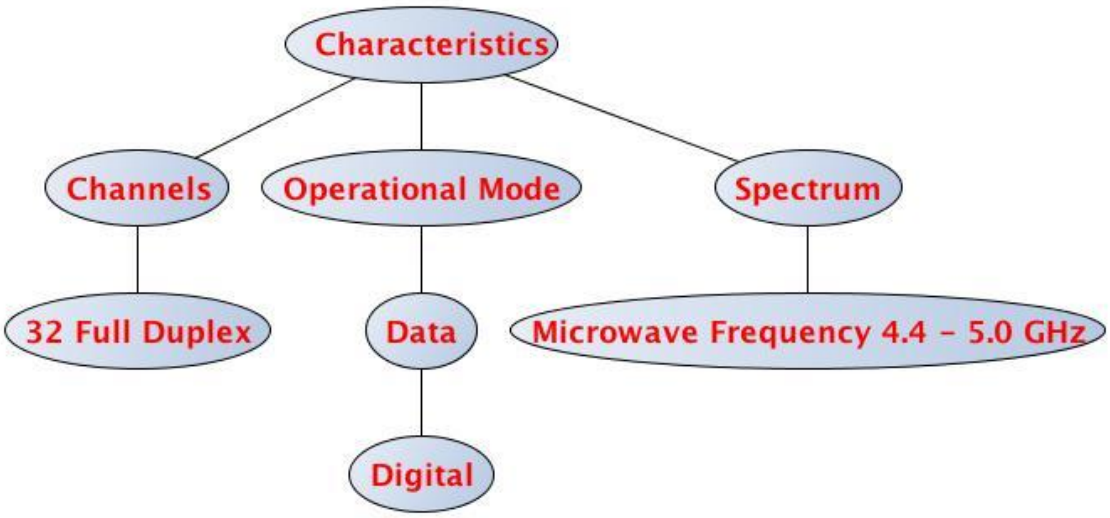


Figure 33. Network Planning and Operations Taxonomy (Common Characteristics)

As an extension to our taxonomy, the characteristic data in Figure 33 could be expanded to provide additional information such as range, power, maximum data rates, and channel capacity, etc., further enhancing the overall picture. This taxonomy is fully extensible; it can be expanded to show capabilities and limitations, represent other domains within Network Planning and Operations, and link to other non-communications-oriented related domains, such as Logistics, providing another level of interoperability.

F. CONCLUSION

An information-oriented approach that uses technology enablers like XML and XSLT will help us achieve interoperability and is a solid foundation for SOA and cloud computing. It considers the guidance and future growth of DoD and avoids many of the common pitfalls of SOA adoption by using the industry standard XML, which is also an acceptable common format to allow applications within or between enterprises to exchange information. Additionally, it provides some immediate results such as being loosely coupled; meaning that its designed and developed code is easily adaptable to new devices/applications and it requires limited and/or no development effort for new devices/applications. XSLT on the other hand, provides a standard XML document transformation mechanism that ensures a common language for transformation, a common standard for representing transformation behaviors, and a common input/output document structure. The use of a repository will enable us to join equivalent data and maintain application specific rules. The network planning and operations taxonomy presented serves as a means to achieve interoperability between the network planning, analysis, and monitoring systems. Together, through the use of XML, XSLT, a centralized repository, and a common understanding of the domain, we can provide a feasible and acceptable means of achieving interoperability within the domain of network planning and increase efficiency through the reduction of duplicate, manual processes. The next section validates our assumptions through analysis and a detailed proof-of-concept.

THIS PAGE INTENTIONALLY LEFT BLANK

V. PROOF OF CONCEPT OVERVIEW

A. INTRODUCTION

This chapter provides an overview of the proof of concept application that was developed in conjunction with this thesis. The primary goal of the proof of concept was to successfully implement the software design patterns and techniques presented in Chapters II and III in the form of a working model. The result was a cloud-based SaaS application capable of demonstrating the ability for communications planners to collaborate in near-real time in the communications planning process. It also demonstrated the ability to enable interoperability between the CCP application and other communications planning and monitoring applications using XSL transformations.

B. TECHNOLOGIES USED

1. Programming Environment

Being a web browser based application, CCP relies heavily on Javascript for the provision of AJAX services. In order to develop the application within the time constraints given, we decided on the open source product Google Web Toolkit (GWT) [32] as an appropriate development framework. GWT provided several benefits, including:

- The GWT software development kit is provided as a plug-in to the open-source Eclipse Integrated Development Environment (IDE).
- The entire application can be written in a single programming language (Java). The GWT compiler compiles client-side code into Javascript that is already optimized for efficiency and cross-browser compatibility. Server-side code is packaged in a standard Java WAR file, ready for deployment on the Java web server of choice.
- GWT provides built-in client server communication via remote proxy, reducing the amount of time required for programming this interface.

2. Server Environment

- Ubuntu 10.10 with Linux kernel 2.6.35
- Apache HTTP server 2.2.16
- Apache Tomcat server 6.0.28
- Java Virtual Machine 1.6
- MySQL client version 5.1.61

C. APPLICATION FEATURES

The main client screen (Figure 34) presents a modularized view of all aspects of the project to the user. Each of these view modules is highlighted below. Because of the modular nature of the design, the ability to move, replace, or add a view is trivial, with the final design decision left to the stakeholders. The network diagram canvas area displays Sites, Links and Circuits in the standard format used by Marine Corps communications planners. Each element can be repositioned within the view based on the planner's preferences by dragging the element with the mouse. Sites can also be resized using the mouse. Connector lines and Link information boxes are automatically drawn and updated as required. All changes made by a planner on one client are propagated to the server, which updates all other connected clients in near-real time (as described in Chapter II).

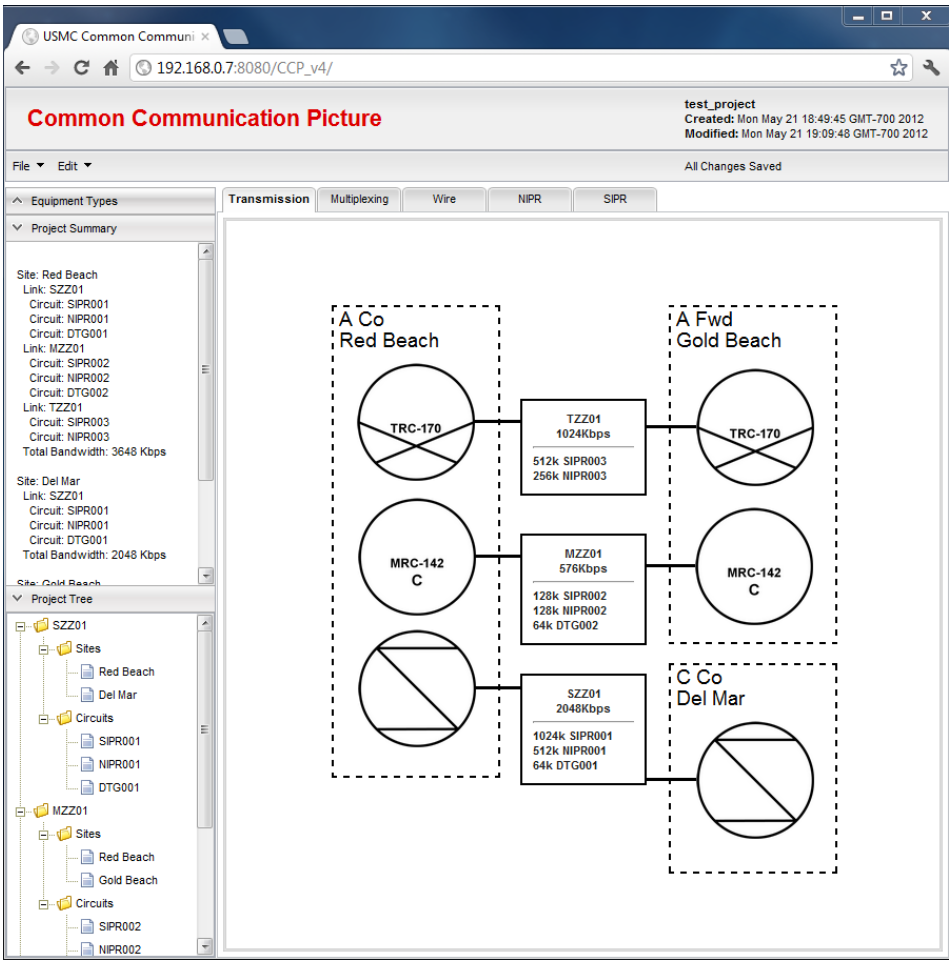


Figure 34. CCP main project editing window

1. Network Diagram Tabs

Each Circuit type requires its own diagram (i.e., all NIPR circuits will be represented on the NIPR diagram). Tabs provide a simple and logical way to present the various network diagrams associated with a given project (Figure 35). Tabs also assist in enabling multiple planners edit different diagrams simultaneously.

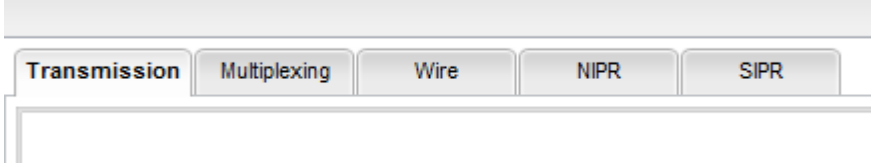


Figure 35. Network diagram tabs

2. Global and Specialized Menus

Menus are the main method of interacting with the user interface to initiate a change. A global menu is provided via the menu bar (Figure 36).

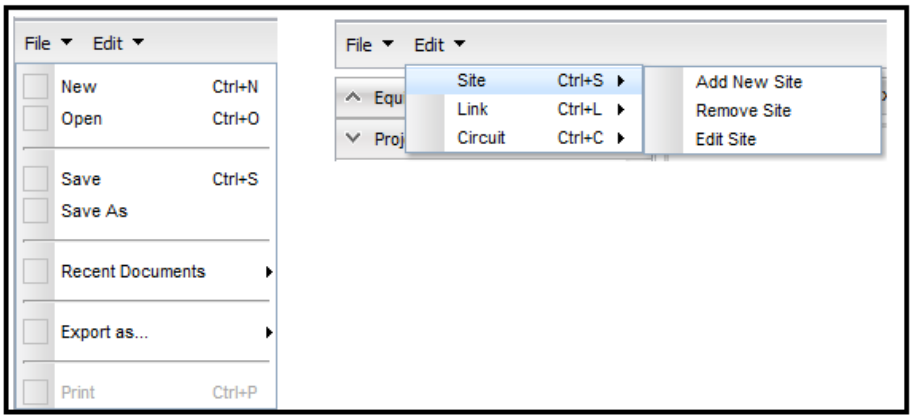


Figure 36. Menu bar File and Edit menus

Element-specific menus are provided for Site, Link and Circuit network entities. They are available by right clicking with the mouse on the entities respective GUI element (Figure 37). Selection of a menu option will result in the appropriate form being displayed for use action.

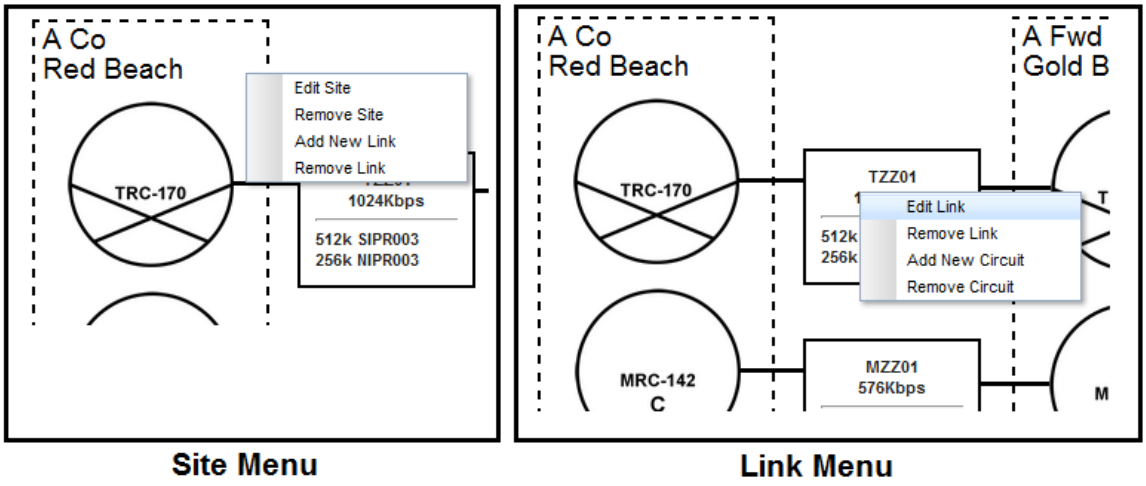


Figure 37. Site and Link right-click menus

3. Project Tree

An expandable tree chart enables a simplified overview of the project (Figure 38).

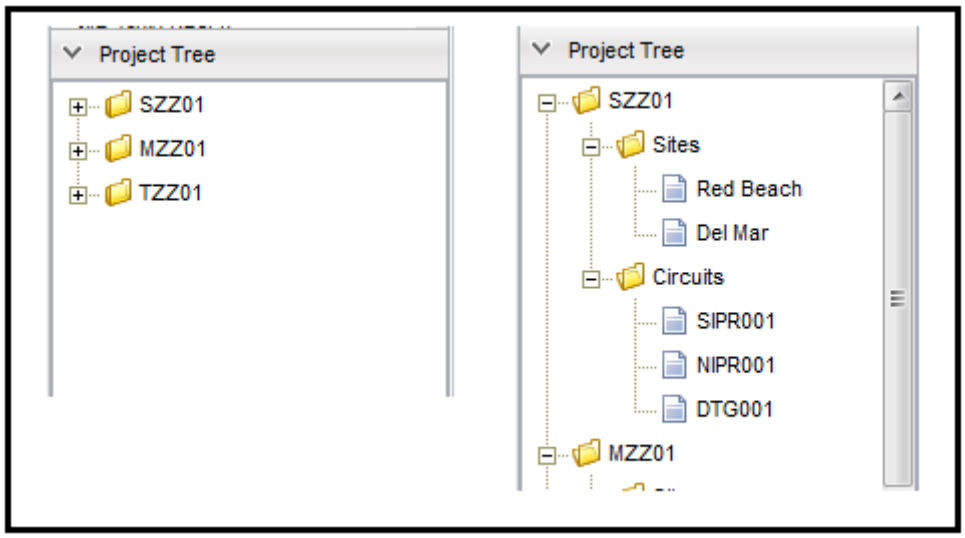


Figure 38. Expandable project tree

4. Project Information Area

The information area displays the project's name, creation date, and the date and time that the project was last modified (Figure 39). It also provides confirmation that any changes made by the planner have been successfully transmitted to the server by noting "All Changes Saved." This message reads "Saving Changes…" when an update is in progress.

Figure 39. Project information area

5. Forms

Forms provide the interface for creating and editing project elements (Figure 40).

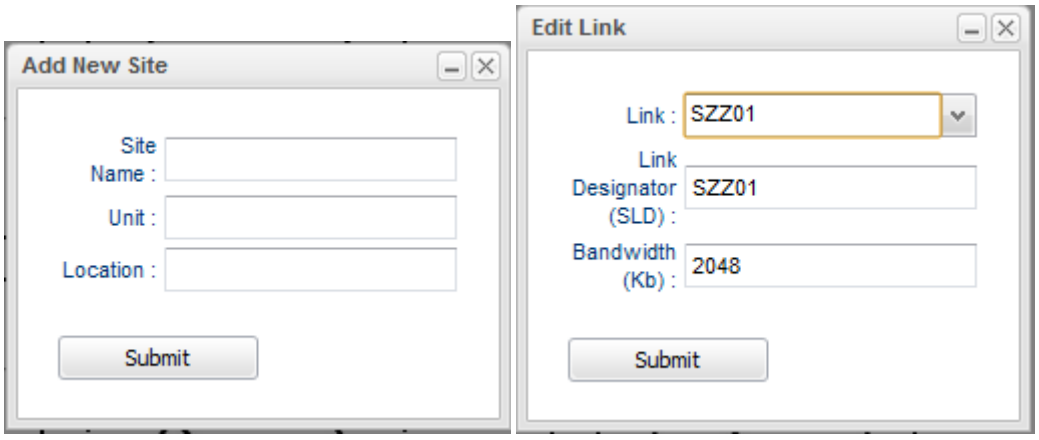


Figure 40. Sample CCP forms

6. Project Export

Selecting the “Save As” menu option from the menu bar enables the user to save a local copy of the project in XML format as shown in Figure 41.

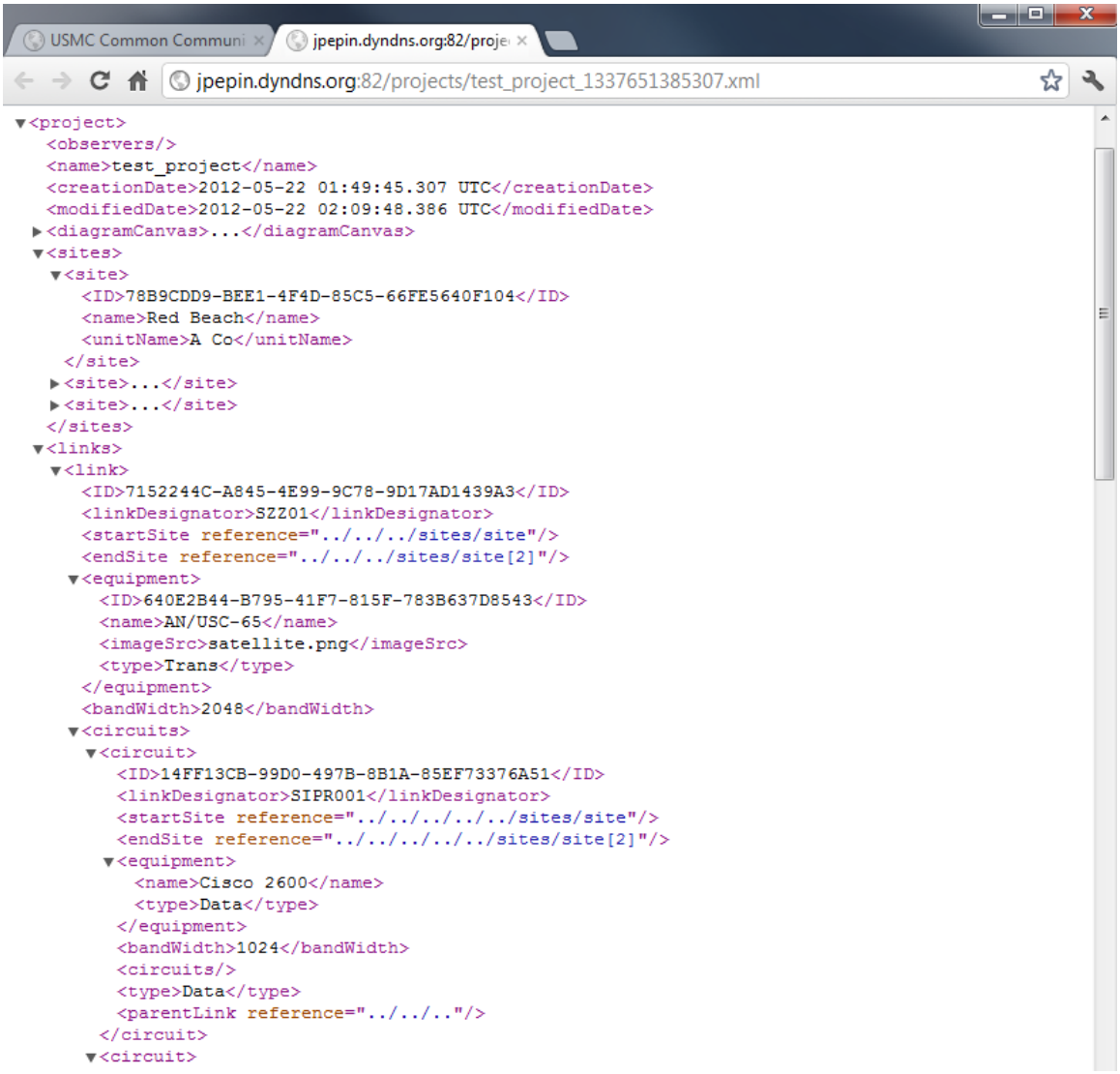


Figure 41. Sample project XML output

The “Export” option in the global menu is meant to provide a means for the user to download the project information in pre-formatted form to be imported into a third-party application. For our proof of concept, we chose to demonstrate this by converting the CCP XML format to SPEED XML format. The user was able to import the resultant XML file into SPEED, removing the work involved in manually re-entering redundant network information. While demonstrated using SPEED, this capability is extensible to other applications that are able to import information in XML format.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

A. SUMMARY OF WORK

Communications planning is an iterative, time-dependent process that benefits greatly from technologies that enable collaboration between planners and information sharing between planning and monitoring applications. Current collaboration techniques have undoubtedly enhanced the planning process, and have allowed planners to meet the high demands of the modern-day operating environment. However, an examination of the current planning workflow reveals that there is still much room for improvement. By reducing redundant tasks and automating processes, as demonstrated in this thesis, the time required to complete the planning cycle can be further reduced, and the end result more accurate.

This thesis identified inefficiencies in the current communication planning process that can be resolved using a cloud-based application. Tools and techniques for designing and developing web browser-based applications have advanced to the point that their capabilities rival equivalent standalone desktop applications. By making deliberate software design decisions, we can enable near real-time planning and collaboration in a ubiquitous environment, independent of user platform or geographic location. Additionally, the ability of applications to interoperate through the enabling of information sharing further reduces the amount of time required to complete the planning cycle by eliminating redundant tasks. We have shown that, when done correctly, cloud-based communications planning is not only possible, but is advantageous, and the changes in workflow that occur can result in a more efficient communications planning process.

B. CONTRIBUTIONS

The main contribution of this thesis is a high-level design for a cloud-based application called the Common Communication Picture (CCP). As

demonstrated in the proof of concept, the CCP application enables near-real time collaboration between communications planners, and provides a bridge for enabling interoperability between communications planning and monitoring tools. This thesis provides analysis and recommendations for the use of existing software design patterns. It presents variations in the implementation of these patterns that result in more efficient network usage (e.g., the partial DTO pattern), a characteristic which is critical of network software operating on tactical networks. It also introduces a new pattern, called the Observer Manager, which extends the well-known Observer pattern, resulting in a more loosely coupled and modular software design. Finally, this thesis analyses the ability to enable interoperability between existing and future communications planning and monitoring tools, and provides a feasible technical solution using XML and XSLT.

C. RECOMMENDATIONS FOR FUTURE WORK

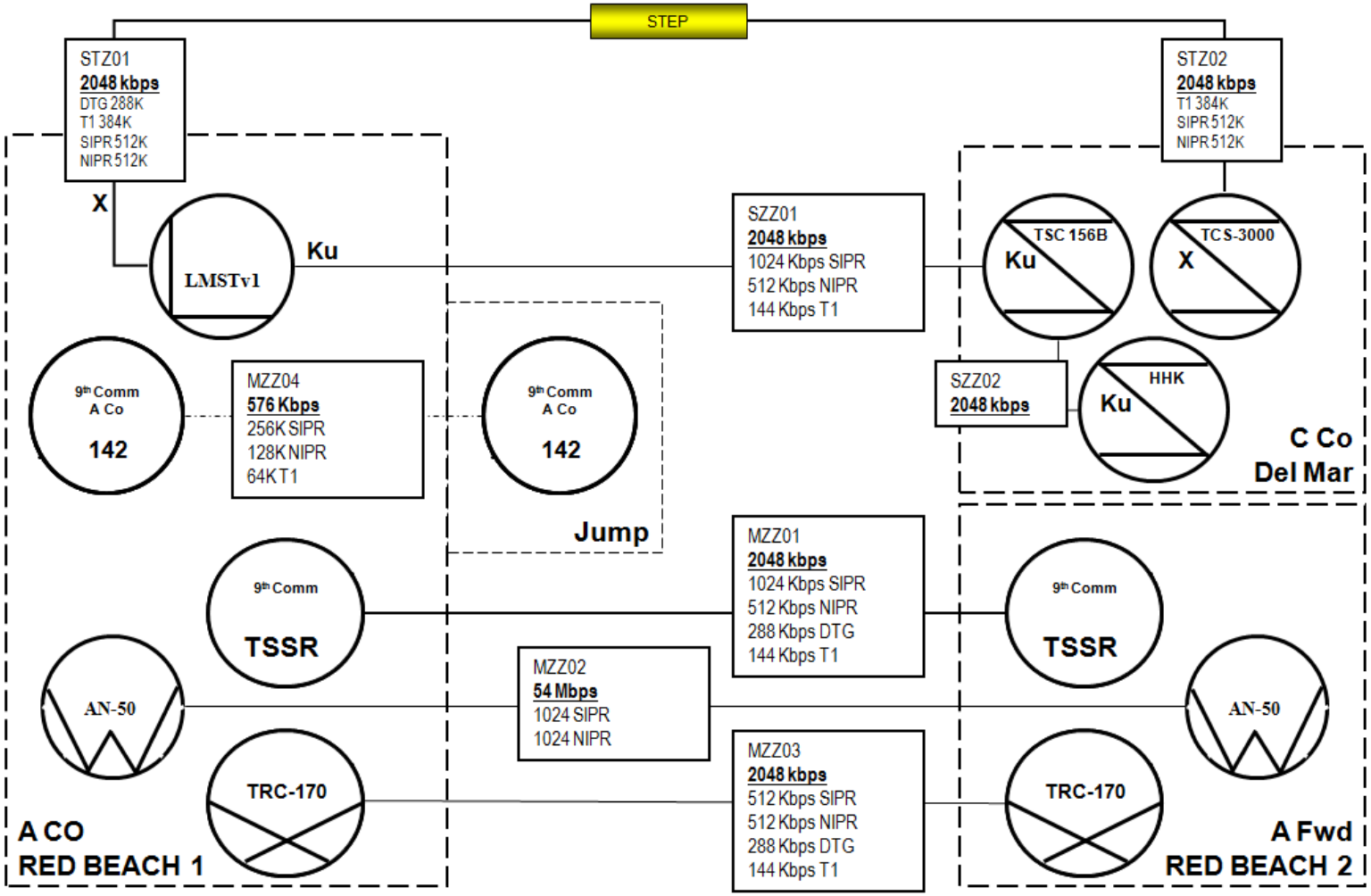
The high-level design presented in this thesis provides a foundation on which to develop a fully capable system through the iterative development process resulting in a working prototype that can be thoroughly tested and evaluated by organizations that may be interested in the resultant capability, such as MCTSSA and MARCORSYSCOM. The following are some recommendations for future work in the development of a cloud-based communication planning tool:

- Conduct real-world usability and feasibility testing of a working prototype with a representative subset of communications planners. Incorporation of their feedback and recommendations is crucial to ensure success in the operational environment.
- Project what the scalability requirement will be for CCP. Determining a realistic upper bound on the number of simultaneous planners will assist in determining server requirements in real-world operation, and could result in changes to design pattern recommendations in order to provide greater scalability.

- Incorporate a centralized equipment database. Currently, the proof of concept application uses hard-coded equipment specifications to demonstrate the ability to automate diagram creation. In order to provide the level of interoperability and error checking required of a fully-functional system, CCP should be modified or extended to access a centralized database during the project editing process.
- Access to, and incorporation of data from, the GSORTS database. This will further reduce errors in the planning process by ensuring that only equipment that belongs to a specific unit, is currently available and is in an operational status are incorporated in a network plan.
- Recommended additional CCP features:
    - Add the ability for users to upload reference documents and associate them with a project.
    - Add ability to import from file. Currently, CCP can only export a project in XML format, but not import. Incorporating this feature will allow users to create a new project from an existing project by importing an XML file.
- Extend CCP to provide information (such as documents and application specific XML files) in the form of Web services to current and future communication planning and monitoring applications.
- Extend CCP to provide a mobile user interface, in order to enable referencing of diagrams and documents by network installers and operators on a mobile device in the field.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: SAMPLE TRANSMISSION DIAGRAM



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: CCP USE CASE DESCRIPTIONS

A. UC-1: CREATE NEW PROJECT

Actors

Lead Communications Planner (Primary), Server Administrator

Description

The lead communications planner logs into the Common Communications Picture (CCP) planning tool and creates a new communications network project via options in the user interface. The lead planner assigns the project a name, and identifies users who are able to edit the project (or a subsystem therein). During project creation, the lead planner specifies the types of networks to be planned (data, wire, single-channel radio, multi-channel radio, etc.) based on the mission requirements and equipment availability of an operation or exercise. He is able to create initial network diagram frameworks for each subsystem through the project creation dialog for later editing by him or other communications planners that have permission to edit the project (or a given subsystem). As required, the server administrator can perform the initial steps of creating a new project and assigning the appropriate level of permission for editing and sharing to the planners involved.

B. UC-2: EDIT EXISTING PROJECT

Actors

Lead Communications Planner (Primary), Communications Planner

Description

Editing a plan involves adding and removing network entities and changing allocated resources (bandwidth, frequencies, block of IP addresses or phone numbers, etc.). Planners select an entity (site, link or circuit) from a selection box to add to a diagram, and can connect it to an existing object in the

diagram. The planning tool will display an error if a connection between two objects cannot be made due to a conflict. Any resource shared between the two objects, such as bandwidth, is annotated appropriately. On completion of editing, the planner can choose to save the plan or exit, discarding changes. If saved, the edited plan becomes the “current” version. The previous “current” version is still viewable at a later date. All saved previous versions of the plan can be viewed as needed by planners. Previous versions can also be deleted by planners with the appropriate permission level.

C. UC-3: IMPORT FROM THIRD-PARTY APPLICATION

Actors

Lead Communications Planner (Primary), Communications Planner

Description

The system allows for the importing of XML-formatted text files that have been exported from supported third-party applications (such as SPEED). The planner selects the option to import a file from the application menu, indicating the source application. Once the selected file has been uploaded to the server, the CCP application converts the contents of the imported file into CCP format and displays the results to the planner as a newly created Project.

D. UC-4: EXPORT TO THIRD-PARTY APPLICATION

Actors

Lead Communications Planner (Primary), Communications Planner

Description

The system allows for the exporting of a text file pre-formatted so that it can be imported into a supported third-party application (such as SPEED). The planner selects the option to export a file from the application menu, indicating the target application format. The CCP application creates an XML-formatted file

and populates it with relevant information from the current Project. The file is downloaded to the planner's local computer for import into the target application.

E. UC-5: VIEW NETWORK DIAGRAMS

Actors

Lead Communications Planner (Primary), Communications Planner, End User

Description

The end result of the planning process is a set of network diagrams that can be used by network installers and maintainers (end users in this case) in the execution of their assigned tasks. To view the diagrams for a plan, the end user must have the appropriate permission level. After logging in to the system through the log-in interface, all projects for which a user has permissions are displayed. The user can select the appropriate project from those displayed, and view all associated diagrams. Other end users are those persons or units that have a need to know the current network topology, such as a unit training to take over the network. Only the most recent network diagrams are displayed for the end users with view-only permission. Viewers with the proper permission level can write comments pertaining to the project for other users to read.

F. UC-6: SET USER PERMISSIONS

Actors

Server Administrator (Primary), Lead Communications Planner

Description

There are multiple levels of permission, ranging from not being able to view a project, to the ability to create and edit a project. The server administrator has the ability to assign permissions at all levels, usually at account creation time. The lead planner is one who has the ability to create a new project. Once a project has been created (UC-1), the project can be edited by all users at the

appropriate permission level, and saved for later viewing or further editing. The lead planner specifies, via a user permissions dialogue, whether a subordinate planner is able to edit the whole plan, or a portion thereof. Communications planners have the ability to edit only those portions of the project that their permissions allow.

G. UC-7: ADD/REMOVE USER ACCOUNTS

Actors

Server Administrator (Primary)

Description

The server administrator has the ability to add and remove user accounts through the administrator interface. Requests for new accounts are sent by users via a request dialog on the tool's main user interface. The server administrator, upon validating the request, sets the appropriate permission level for the user. Likewise, the server administrator has the ability to remove a user account through the same interface. Both transactions involve manipulating data stored in the CCP back-end database.

H. UC-8: EDIT EQUIPMENT DATABASE

Actors

Database Administrator (Primary)

Description

The system allows for the addition, deletion and modification of existing hardware specifications (such as the ability to communicate with a new system). The database administrator edits an existing piece of equipment through the database administration interface. This interface communicates with the server's database to store the edited hardware specification. Specifications include hardware identification number, hardware compatibility, bandwidth limits, capabilities and limitations (such as number of interfaces).

APPENDIX C: THE PROJECT CLASS

```
package com.pepinonline.ccp.shared.project;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

import com.pepinonline.ccp.shared.IsObservable;
import com.pepinonline.ccp.shared.util.Printer;

public class Project extends IsObservable implements Serializable {

	/**
	 * Need to Serialize Object to save to file, and to pass
	 * over RPC.
	 */
	private static final long serialVersionUID = 1L;

	private static Project currentProject = null;
	private String name;
	private Date creationDate;
	private Date modifiedDate;
	private String diagramCanvas;

	/**
	   * This field is a List that must always contain Sites.
	   *
	   * @gwt.typeArgs <Site>
	   */
	private List<Site> sites;

	/**
	   * This field is a List that must always contain Links.
	   *
	   * @gwt.typeArgs <Link>
	   */
	private List<Link> links;


	//------------- CONSTRUCTORS --------------//

	private Project() {
		// required for GWT
	}

	private Project(String name) {
		this.name = name;
		this.creationDate = new Date();
		this.modifiedDate = this.creationDate;
		this.sites = new ArrayList<Site>();
```

```
            this.links = new ArrayList<Link>();
      }

      /**
       * Returns the singleton object, but does not
       * create on if it doesn't exist.
       * @return
       */
      public static Project getProject() {
            return currentProject;
      }

      /**
       * Singleton object initialized lazily, on first demand.
       * Synchronized singleton initializer prevents creation of
multiple
       * instances of class object if called by two clients at the
       * same time.
       * @param name
       * @return
       */
      public static synchronized Project getProject(String name) {
            // if the project exist
            if (currentProject == null) {
                  Printer.printDebug("Creating new Project object");
                  currentProject = new Project(name);
            }
            return currentProject;
      }


      //--------- Setters ------------------

      public void setName(String name) {
            this.name = name;
            this.modifiedDate = new Date();
            notifyObservers(false);
      }

      public void updateModifiedDate() {
            this.modifiedDate = new Date();
      }

      public void updateDiagramCanvas(String diagramDetails) {
            diagramCanvas = diagramDetails;
            this.modifiedDate = new Date();
            updateProjectDTO(1);
            notifyObservers(false);
            Printer.printDebug("updateDiagramCanvas client: " +
                        this.modifiedDate.hashCode());
      }

      //--------- Getters ------------------

      public String getName() {
```

```
      return this.name;
}

public Date getCreationDate() {
      return this.creationDate;
}

public Date getModifiedDate() {
      return this.modifiedDate;
}

public List<Site> getSites() {
      return this.sites;
}

public List<Link> getLinks() {
      return this.links;
}

public String getDiagramCanvas() {
      return diagramCanvas;
}

//------- Other modifier methods ---------------

public void addSite(Site siteName) {
      Printer.printDebug("Project.addSite site: " +
                  siteName.toString());
      // add site to sites list if not already on it
      if (!sites.contains(siteName)){
            sites.add(siteName);
            updateModifiedDate();
            updateProjectDTO(0);
            notifyObservers(false);
      }
}

public void addLink(Link newLink) {
      // add link to links list if not already on it
      if (!links.contains(newLink)){
            links.add(newLink);
            updateModifiedDate();
            updateProjectDTO(0);
            notifyObservers(false);
      }
}

public void removeSite(Site siteName) {
      Printer.printDebug("Project.removeSite site: " +
                  siteName.toString());
      if(sites.remove(siteName)) {
            // remove all links associated with the site
            // being removed
            Iterator<Link> iter = links.iterator();
            while (iter.hasNext()) {
```

```
                        Link l = iter.next();
                        if (l.getStartSite() == siteName ||
                                    l.getEndSite() == siteName) {
                              iter.remove();
                        }
                  }
                  updateModifiedDate();
                  updateProjectDTO(0);
                  notifyObservers(false);
            }
      }

      public void removeLink(Link linkName) {
            links.remove(linkName);
            updateModifiedDate();
            updateProjectDTO(0);
            notifyObservers(false);
      }

      //---------- DTO Methods -------//

      public ProjectDTO getCompleteDTO() {
            return toCompleteDTO();
      }

      private ProjectDTO toCompleteDTO() {
            ProjectDTO dto = ProjectDTO.getDto();
            dto.setName(this.name);
            dto.setCreationDate(this.creationDate);
            dto.setModifiedDate(this.modifiedDate);
            dto.setSites(this.sites);
            dto.setLinks(this.links);
            dto.setDiagramCanvas(this.diagramCanvas);
            return dto;
      }

      public void fromDtoUpdate(ProjectDTO dto) {
            // update Project fields with non-null DTO values
            if (dto.getName() != this.name) {
                  this.name = dto.getName();
            }
            if (dto.getCreationDate() != null) {
                  this.creationDate = dto.getCreationDate();
            }
            if (dto.getModifiedDate() != this.modifiedDate) {
                  this.modifiedDate = dto.getModifiedDate();
            }
            if (dto.getSites() != null) {
                  Printer.printDebug("fromDtoUpdate sites updated from
DTO");
                  this.sites = dto.getSites();
            }
            if (dto.getLinks() != null) {
                  Printer.printDebug("fromDtoUpdate links updated from
DTO");
```

```
                this.links = dto.getLinks();
            }
            if (dto.getDiagramCanvas() != null) {
                Printer.printDebug("fromDtoUpdate diagramCanvas
updated                          from DTO");
                this.diagramCanvas = dto.getDiagramCanvas();
            }
            notifyObservers(true);
      }

      /**
       * Create new Project object from an existing ProjectDTO
       * @param proxy
       */
      public static Project fromDtoNew(ProjectDTO dto) {
            if (currentProject == null) {
                currentProject = new Project();
            }
            currentProject.fromDtoUpdate(dto);
            return currentProject;
      }

      /**
       * Update the DTO based on the element that was changed.
       * Options are: 0 = the project object was modified, such
       * as adding or removing a site or link. 1 = the diagramCanvas
       * was modified, as in changing the canvas location of an
element.
       * The result is a DTO that contains only the pertinent data
       * that can be sent more efficiently.
       *
       * @param elementChanged - Flag indicating which part of the
Project
       * has been modified
       */
      public void updateProjectDTO(int elementChanged) {
            ProjectDTO dto = ProjectDTO.getDto();
            // empty all current DTO fields
            dto.clear();
            dto.setName(currentProject.getName());
            dto.setModifiedDate(currentProject.getModifiedDate());
            switch(elementChanged) {
            // must update both sites and links arrays in order
            // to maintain proper link-site pointers
            case 0: dto.setSites(currentProject.getSites());
                    dto.setLinks(currentProject.getLinks());
                    Printer.printDebug("updateProjectDTO: DTO sites and
                              links updated");
                  // don't break because a change in sites or links
                  // necessitates a change in the diagram that depicts
                  // the project
            case 1:
dto.setDiagramCanvas(currentProject.getDiagramCanvas());
                    Printer.printDebug("updateProjectDTO: DTO
                              diagramCanvas updated");
```

```
                        break;
                }
        }

        /**
         * Check to see if the project contains a site with the given
name.
         */
        public boolean containsSite(String siteName) {
                for (Site s: this.sites) {
                        if (s.getName().equals(siteName)) {
                                return true;
                        }
                }
                return false;
        }

        /**
         * Check to see if the project contains a link
         * with the given designator.
         */
        public boolean containsLink(String linkDesignator) {
                for (Link l: this.links) {
                        if (l.getLinkDesignator().equals(linkDesignator)) {
                                return true;
                        }
                }
                return false;
        }

}
```

APPENDIX D: THE PROJECT DTO CLASS

```
package com.pepinonline.ccp.shared.project;

import java.io.Serializable;
import java.util.Date;
import java.util.List;

public class ProjectDTO implements Serializable {

	private static final ProjectDTO projectDto = new ProjectDTO();

	/**
	 * Needed to Serialize Object to save to file, and to pass
	 * over RPC.
	 */
	private static final long serialVersionUID = 1L;
	private String name;
	private Date creationDate;
	private Date modifiedDate;

	/**
	 * String representation of diagram
	 */
	private String diagramCanvas;

	/**
	   * This field is a List that must always contain Sites.
	   *
	   * @gwt.typeArgs <Site>
	   */
	private List<Site> sites;

	/**
	   * This field is a List that must always contain Links.
	   *
	   * @gwt.typeArgs <Link>
	   */
	private List<Link> links;

	private ProjectDTO() {
		// required for GWT
	}

	public static ProjectDTO getDto() {
		return projectDto;
	}

	//--------- Setters ------------------

	public void setName(String name) {
		this.name = name;
	}
```

```
public void setCreationDate(Date created) {
      this.creationDate = created;
}

public void setModifiedDate(Date modified) {
      this.modifiedDate = modified;
}

public void setSites(List<Site> sites) {
      this.sites = sites;
}

public void setLinks(List<Link> links) {
      this.links = links;
}

public void setDiagramCanvas(String diagramDetails) {
      this.diagramCanvas = diagramDetails;
}

//--------- Getters -----------------

public String getName() {
      return this.name;
}

public Date getCreationDate() {
      return this.creationDate;
}

public Date getModifiedDate() {
      return this.modifiedDate;
}

public List<Site> getSites() {
      return this.sites;
}

public List<Link> getLinks() {
      return this.links;
}

public String getDiagramCanvas() {
      return diagramCanvas;
}

/**
 * Nullify all DTO fields.
 */
public void clear() {
      this.name = null;
      this.creationDate = null;
      this.modifiedDate = null;
      this.sites = null;
```

```
            this.links = null;
            this.diagramCanvas = null;
        }

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E: THE OBSERVER MANAGER CLASS

```
package com.pepinonline.ccp.shared;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import com.pepinonline.ccp.shared.IsObservable;
import com.pepinonline.ccp.shared.Observer;
import com.pepinonline.ccp.shared.util.Printer;

public class ObserverManager extends IsObservable implements
Serializable, Observer {

	private static final long serialVersionUID = 1L;

	private static ObserverManager manager;

	private LinkedHashMap<IsObservable, List<Observer>> observerMap =
		new LinkedHashMap<IsObservable, List<Observer>>();

	private ObserverManager() {
		// Singleton object
	}

	public static ObserverManager getObserverManager() {
		if(manager == null){
			manager = new ObserverManager();
		}
		return manager;
	}

	public void replaceObservedObject(IsObservable oldObject,
			IsObservable newObject) {
		List<Observer> observers = observerMap.get(oldObject);
		observerMap.put(newObject, observers);
		observerMap.remove(oldObject);
	}

	//-------------- Extends IsObservable ----------------------//

	@Override
	public void addObserver(IsObservable i, Observer o) {
		if(observerMap.get(i) != null) {
			// if the IsObservable is already being observed,
			// add observer to its mapped List of observers.
			observerMap.get(i).add(o);
		} else {
			// if the IsObservable is not already being observed,
			// make the observer manager an observer of the new
				// IsObservable
			i.addObserver(this);
```

```
                  // create a list of observers to map to the
isObservable
                  List <Observer> temp = new ArrayList<Observer>();
                  // add the Observer parameter as the first list item
                  temp.add(o);

                  // create the map entry
                  observerMap.put(i, temp);
            }
      }

      @Override
      public void removeObserver(IsObservable i, Observer o) {
            if(observerMap.get(i) != null) {
                  observerMap.get(i).remove(o);
            }
      }

      @Override
      public void notifyObservers (IsObservable i, boolean remote) {
            // Notify each observer for this IsObservable Object
        List<Observer> observers = observerMap.get(i);
        if (observers == null || observers.isEmpty()){
            return;
        }
        for (Observer o: observers) {
            try{
                  o.update(i, remote);
            } catch (Exception e){
                  Printer.printDebugErr("ObserverManager
notifyObservers: "
                        + e.getMessage());
            }
        }
      }

      //-------------- Implements Observer ----------------------//

      @Override
      public void update(IsObservable i, boolean remote) {
            notifyObservers(i, remote);
      }

}
```

APPENDIX F: THE ISOBSERVABLE CLASS

```
package com.pepinonline.ccp.shared;

import java.util.ArrayList;
import java.util.List;

public abstract class IsObservable {

	private List<Observer> observers = new ArrayList<Observer>();

	public void addObserver(Observer o) {
		observers.add(o);
	}

	public void addObserver(IsObservable io, Observer o) {
		if (io.equals(this)) {
			observers.add(o);
		}
	}

	public void removeObserver(Observer o) {
		observers.remove(o);
	}

	public void removeObserver(IsObservable io, Observer o) {
		if (io.equals(this)) {
			observers.remove(o);
		}
	}

	public void notifyObservers(boolean remote) {
		for (Observer o: observers) {
			o.update(this, remote);
		}
	}

	public void notifyObservers(IsObservable io, boolean remote) {
		if (io.equals(this)) {
			for (Observer o: observers) {
				o.update(this, remote);
			}
		}
	}
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G: THE OBSERVER INTERFACE

```
package com.pepinonline.ccp.shared;

public interface Observer {

	public void update( IsObservable o, boolean remote );

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H: CCP SCHEMA

```
<?xml version="1.0" encoding="utf-16"?>

<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="project">

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="observers" type="xsd:string" />

        <xsd:element name="name" type="xsd:string" />

        <xsd:element name="creationDate" type="xsd:string" />

        <xsd:element name="modifiedDate" type="xsd:string" />

        <xsd:element name="diagramCanvas" type="xsd:string" />

        <xsd:element name="sites">

          <xsd:complexType>

            <xsd:sequence>

              <xsd:element maxOccurs="unbounded" name="site">

                <xsd:complexType>

                  <xsd:sequence>

                    <xsd:element name="ID" type="xsd:string" />

                    <xsd:element name="name" type="xsd:string" />

                    <xsd:element name="unitName" type="xsd:string" />

                  </xsd:sequence>

                </xsd:complexType>
```

```
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="links">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="link">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="ID" type="xsd:string" />
            <xsd:element name="linkDesignator" type="xsd:string" />
            <xsd:element name="startSite">
              <xsd:complexType>
                <xsd:attribute name="reference" type="xsd:string" />
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="endSite">
              <xsd:complexType>
                <xsd:attribute name="reference" type="xsd:string" />
              </xsd:complexType>
            </xsd:element>
```

```
<xsd:element name="startLocation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="latitude" type="xsd:decimal" />
      <xsd:element name="longitude" type="xsd:decimal" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="endLocation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="latitude" type="xsd:decimal" />
      <xsd:element name="longitude" type="xsd:decimal" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="equipment">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ID" type="xsd:string" />
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="imageSrc" type="xsd:string" />
      <xsd:element name="type" type="xsd:string" />
```

```
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="bandWidth" type="xsd:int" />
                <xsd:element name="circuits" type="xsd:string" />
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

LIST OF REFERENCES

[1] United States Marine Corps, MCWP 5–1 Marine Corps Planning Process (w/Chg 1), 2010, pp. 1–5 - 1–6.

[2] United States Marine Corps, MCWP 3–40.3 MAGTF Communications System, 8 January 2010, pp. 6–1.

[3] MARCORSYSCOM, “Systems Planning Engineering and Evaluation Device (SPEED)l,” [Online]. Available: http://www.marcorsyscom.usmc.mil/sites/cins/NSC/TEAM%20EXPEDITIONARY%20COMMUNICATIONS/SPEED.html.

[4] Defense Information Systems Agency, “Joint Communication Simulation System,” [Online]. Available: http://www.disa.mil/jcss/about.html.

[5] SolarWinds, “Solarwinds Orion,” [Online]. Available: http://www.solarwinds.com/products/network-management/network-monitoring.aspx.

[6] IpSwitch, “WhatsUp Gold,” [Online]. Available: http://www.whatsupgold.com.

[7] United Stated Marine Corps, “Marine Corps Warfighting Publication (MCWP) 3–40.3,” pp. 6–5.

[8] CJCS, *CJCSI 3401.02A, GLobal Status of Resources and Training System (GSORTS),* 2008.

[9] United Stated Marine Corps, “Marine Corps Order 3000.13,” in *Marine Corps Reporting Standard Operating Procedures (SOP)*, 30 July 2010, pp. 1–2.

[10] V. Kundra. (8 February 2011). “Federal Cloud Computing Strategy,” [Online]. Available http://www.cio.gov.

[11] United States Marine Corps, in *MCWP 6–22 Communications and Information Systems*, 16 November 1998, pp. 6–2.

[12] United Stated Marine Corps, in *MCWP 3–40.3 MAGTF Communications Systems*, 8 January 2010, pp. 2–4.

[13] United Stated Marine Corps, in *MCWP 3–40.3 MAGTF Communications Systems*, 8 January 2010, pp. 5–6.

[14] Oracle, “Concurrency,” [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html.

[15] E. e. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley, 1995.

[16] World Wide Web Consortium (W3C), “The WebSocket API,” [Online]. Available: http://dev.w3.org/html5/websockets/.

[17] Google, “Google MobWrite,” [Online]. Available: http://code.google.com/p/google-mobwrite.

[18] C. Daily, “Comet Maturity Guide” [Online]. Available: http://cometdaily.com/maturity.html.

[19] The Internet Engineering Task Force (IETF), “HTTP/1.1, part 1: URIs, Connections, and Message Parsing” [Online]. Available: http://www.ietf.org/id/draft-ietf-httpbis-p1-messaging-18.txt.

[20] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley Professional, 2002.

[21] Institute of Electrical Engineers, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Glossaries*. IEEE Press, New York, 1990.

[22] Department of Defense, *DoD Directive 5000.01, “The Defense Acquisition System,”* 2007.

[23] R. Tudor, *The Global Information Network Architecture (GINA) Technology Framework*,Dept. Computer Science Naval Postgraduate School, Monterey, CA, Rep. NPS-GSBPP-07-015, 2007.

[24] Department of Defense Chief Information Officer, ““DoD Net Centric Services Strategy: Evolving the DoD Net-Centric Environment to an Enterprise Service Oriented Architecture,” 2006.

[25] D. S. Linthicum, in *Next Generation Application Integration*. Boston, Pearson Education Inc., 2004, pp. 25–53.

[26] D. S. Linthicum, in *Next Generation Application Integration*. Boston, Pearson Education, Inc., 2004, pp. 191–214.

[27] The TEI Consortium, “Text Encoding Initative” [Online]. Available: http://www.tei-c.org/release/doc/tei-p4-doc/html/SG.html#SG17.

[28] Department of Defense, Department of the Navy, “Department of the Navy’s Vision for Extensible Markup Language (XML),” 2002.

[29] Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, “Final Report of teh Defense Science Board Task Force on Achieving Interoperability in a Net-Centric Environment,” 2009.

[30] W3C, “XSL-FO” [Online]. Available: http://www.w3.org/wiki/Xsl-fo.

[31] W3C, “XML Path Language (XPath)” [Online]. Available: http://www.w3.org/TR/xpath/.

[32] Google, “Google Web Toolkit” [Online]. Available: https://developers.google.com/web-toolkit/.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Professor Man-Tak Shing
   Naval Postgraduate School
   Monterey, California

8. Professor John Gibson
   Naval Postgraduate School
   Monterey, California

9. Professor Gurminder Singh
   Naval Postgraduate School
   Monterey, California